The Development of Neural Network Based System Identification and Adaptive Flight Control for an Autonomous Helicopter System

Syariful Syafiq Shamsudin

A thesis presented for the degree of
Doctor of Philosophy
in
Mechanical Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

August 2013

This thesis is dedicated to:
My family for their patience and support during my study.
My friends for brightening my life with their friendship and showing me that life has no greater reward to offer than a true friend.

ABSTRACT

This thesis presents the development of self adaptive flight controller for an unmanned helicopter system under hovering manoeuvre. The neural network (NN) based model predictive control (MPC) approach is utilised in this work. We use this controller due to its ability to handle system constraints and the time varying nature of the helicopter dynamics. The non-linear NN based MPC controller is known to produce slow solution convergence due to high computation demand in the optimisation process. To solve this problem, the automatic flight controller system is designed using the NN based approximate predictive control (NNAPC) approach that relies on extraction of linear models from the non-linear NN model at each time step. The sequence of control input is generated using the prediction from the linearised model and the optimisation routine of MPC subject to the imposed hard constraints. In this project, the optimisation of the MPC objective criterion is implemented using simple and fast computation of the Hildreth's Quadratic Programming (QP) procedure.

The system identification of the helicopter dynamics is typically performed using the time regression network (NNARX) with the input variables. Their time lags are fed into a static feed-forward network such as the multi-layered perceptron (MLP) network. NN based modelling that uses the NNARX structure to represent a dynamical system usually requires a priori knowledge about the model order of the system. Low model order assumption generally leads to deterioration of model prediction accuracy. Furthermore, massive amount of weights in the standard NNARX model can result in an increased NN training time and limit the application of the NNARX model in a real-time application. In this thesis, three types of NN architectures are considered to represent the time regression network: the multi-layered perceptron (MLP), the hybrid

multi-layered perceptron (HMLP) and the modified Elman network. The latter two architectures are introduced to improve the training time and the convergence rate of the NN model. The model structures for the proposed architecture are selected using the proposed Lipschitz coefficient and k-cross validation methods to determine the best network configuration that guarantees good generalisation performance for model prediction.

Most NN based modelling techniques attempt to model the time varying dynamics of a helicopter system using the off-line modelling approach which are incapable of representing the entire operating points of the flight envelope very well. Past research works attempt to update the NN model during flight using the mini-batch Levenberg-Marquardt (LM) training. However, due to the limited processing power available in the real-time processor, such approaches can only be employed to relatively small networks and they are limited to model uncoupled helicopter dynamics. In order to accommodate the time-varying properties of helicopter dynamics which change frequently during flight, a recursive Gauss-Newton (rGN) algorithm is developed to properly track the dynamics of the system under consideration.

It is found that the predicted response from the off-line trained neural network model is suitable for modelling the UAS helicopter dynamics correctly. The model structure of the MLP network can be identified correctly using the proposed validation methods. Further comparison with model structure selection from previous studies shows that the identified model structure using the proposed validation methods offers improvements in terms of generalisation error. Moreover, the minimum number of neurons to be included in the model can be easily determined using the proposed cross validation method. The HMLP and modified Elman networks are proposed in this work to reduce the total number of weights used in the standard MLP network. Reduction in the total number of weights in the network structure contributes significantly to the reduction in the computation time needed to train the NN model. Based on the validation test results, the model structure of the HMLP and modified Elman networks are found to be much smaller than the standard MLP network. Although the total number of weights for both of the HMLP and modified Elman networks are lower than

the MLP network, the prediction performance of both of the NN models are on par with the prediction quality of the MLP network.

The identification results further indicate that the rGN algorithm is more adaptive to the changes in dynamic properties, although the generalisation error of repeated rGN is slightly higher than the off-line LM method. The rGN method is found capable of producing satisfactory prediction accuracy even though the model structure is not accurately defined. The recursive method presented here in this work is suitable to model the UAS helicopter in real time within the control sampling time and computational resource constraints. Moreover, the implementation of proposed network architectures such as the HMLP and modified Elman networks is found to improve the learning rate of NN prediction. These positive findings inspire the implementation of the real time recursive learning of NN models for the proposed MPC controller. The proposed system identification and hovering control of the unmanned helicopter system are validated in a 6 degree of freedom (DOF) safety test rig. The experimental results confirm the effectiveness and the robustness of the proposed controller under disturbances and parameter changes of the dynamic system.

ACKNOWLEDGEMENTS

I believe that I am truly privileged to participate in this fascinating and challenging project as a research member since 2009. I would like to express my deepest gratitude to my supervisor Professor Dr. XiaoQi Chen, who has guided and encouraged my work with such passion and sincerity for knowledge, teaching and care.

I am pleased to acknowledge the financial support from Ministry of Higher Education (MOHE), Malaysia and Universiti Tun Hussein Onn Malaysia (UTHM) under Academic Training Scheme (SLAB) for conference travel fund and scholarship awarded.

I would like to thank my research fellows in Mechatronics Research Laboratory for their help, advice and cooperation for many years. I would like to thank Mervin Chandrapal, Rejina Choi, Stefan Schalk[1], Bob Thijssen[1], Christopher Geelen[1], Julien Pain[2], Aaron Gamble, Peter Tan, Elijah Philips and Bridget Dean for encouraging and supporting my research efforts in LabVIEW® programming, system identification and control theories, unmanned helicopter system integration and test rig system development. I would like to thank the technicians in Mechanical Engineering Department, University of Canterbury, in particular, Julian Murphy and David Read for their invaluable assistance and support to my research project.

My special thanks go to my family. I would like to thank my parents, who taught me to take chances for better things in my life. Also my gratitude to my wife, who has offered me unconditional love, understanding and sacrifices during the period of my study. In retrospect, this project did start humble but has grown to be a success. There are many happy times and many disappointing moments, but now I am very happy because all the hardship I had to go through mostly alone finally paid off.

[1]Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands
[2]Mechanical Engineering Department, INSA de Rouen, France

PUBLICATIONS

The following is a list of my publications that have been published/submitted during the period of my doctoral study:

CONFERENCE PAPERS

- Syariful Syafiq Shamsudin, Xiaoqi Chen, Wenhui Wang, Christopher E. Hann and Geoffrey Chase. 'Neural Networks based System Identification for an Unmanned Helicopter System.' *4^th Asia International Symposium on Mechatronics*, Research Publishing Services: National University of Singapore, Singapore, 2010, pp. 12-19.

- Syariful Syafiq Shamsudin and Xiaoqi Chen. 'Recursive Gauss-Newton Based Training Algorithm for Neural Network Modelling of an Unmanned Helicopter Dynamics.' *19^th International Conference on Mechatronics and Machine Vision In Practice (M2VIP)*, Auckland University of Technology, Auckland, New Zealand, 28-30 November 2012, pp. 92-99.

PEER-REVIEWED JOURNAL PAPERS

- Shamsudin, S.S. and Chen, X.Q. 'Identification of an Unmanned Helicopter System Using Optimised Neural Network Structure', *Int. J. of Modelling, Identification and Control*, 2012, Vol. 17, No. 3, pp. 223–241.

- Shamsudin, S.S. and Chen, X.Q. 'Recursive Gauss-Newton Based Training Algorithm for Neural Network Modelling of an Unmanned Rotorcraft Dynamics', *International Journal of Intelligent Systems Technologies and Applications* , (accepted for publication).

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

AFCS	Automatic Flight Control System
ANN	Artificial Neural Network
AOA	Angle of Attack
ARX	Auto Regressive structure with eXtra inputs
AVCS	Angular Vector Control System
BFGS	Broyden-Fletcher-Goldfarb-Shanno algorithm
BPTT	Back-Propagation Through Time
CG	Centre of Gravity
CIFER®	Comprehensive Identification from Frequency Responses software package
CT	Constant Trace
DBP	Dynamic Back-Propagation
DE	Differential Evolution algorithm
DIO	Digital Input-Output
DOF	Degrees of Freedom
EFRA	Exponential Forgetting and Resetting Algorithm
ESC	Electronic Speed Controller

FIR	Finite Impulse Response
FPGA	Field-Programmable Gate Array
GCS	Ground Control Station
GN	Gauss-Newton
GNC	Guidance, Navigation and Control
HJB	Hamilton-Jacobian-Bellman
HMLP	Hybrid Multi-Layered Perceptron
IMC	Internal Model Controller
IMU	Inertial Measurement Unit
LM	Levenberg-Marquardt
LUT	Look-Up-Table
MIMO	Multiple Inputs-Multiple Outputs
MLP	Multi-Layered Perceptron
MNN	Memory Neural Network
MPC	Model Predictive Control
MSE	Mean Square Error
NBN	Neuron by Neuron algorithm
NI	National Instrument
NMPC	Non-linear Model Predictive Control
NMSS	Non-Minimal State Space
NN	Neural Network
NNAPC	Neural Network based Predictive Control

NNARX	Neural Network ARX
QP	Quadratic Programming
RBF	Radial Basis Function
RC	Remote Control
rGN	Recursive Gauss-Newton method
RHC	Receding Horizon Control
rLM	Recursive Levenberg-Marquardt
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RPE	Recursive Prediction Error method
SAS	Stability Augmentation System
SCARA	Selective Compliant Articulated Robot Arm
SD	Steepest Descent
SISO	Single Input and Single Output
SISO	Single Input-Single Output
SOFM	Self-Organising Feature Map
TITO	Two Inputs-Two Outputs
UART	Universal Asynchronous Receiver/Transmitter
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle

Chapter 1

INTRODUCTION

1.1 BACKGROUND OF THE RESEARCH

An autonomous unmanned aircraft system (UAS) is an instrumented aircraft that is capable of performing given missions autonomously through the use of on-board flight sensors and control system. There has been a substantial increase in number of military and civilian applications that use UAS throughout the past few decades. The increased number of UAS usage is primarily driven by lower risks and higher confidence in mission success associated with UAS utilisation [Kendoul, 2012]. The UAS applications are primarily military related and the main investments are driven by future military usage to integrate UAS into modern military forces. Valavanis [2007] has reported that the civil UAS market will grow slowly over the next decade. Although the civil UAS market growth is predicted to be slower compared with the military market, more government organisations are requiring surveillance and inspection systems for the coast guards, police, border patrol or emergency services for natural disasters. A detailed analysis on the current and predicted UAS market expenditures indicates that the UAS technologies have gained significant importance in the aerospace industry sector where the worldwide UAS market is predicted to expand significantly over the next decade. As an example, research has shown that worldwide UAS market spending will continue to double from the current annual spending of $6.6 billion to $11.4 billion, totalling just over $89 billion in the next ten years [Valavanis, 2007].

The helicopter or rotorcraft based platforms have been among the more popular UAS configuration types used in the aeronautics community. A UAS that is based on a

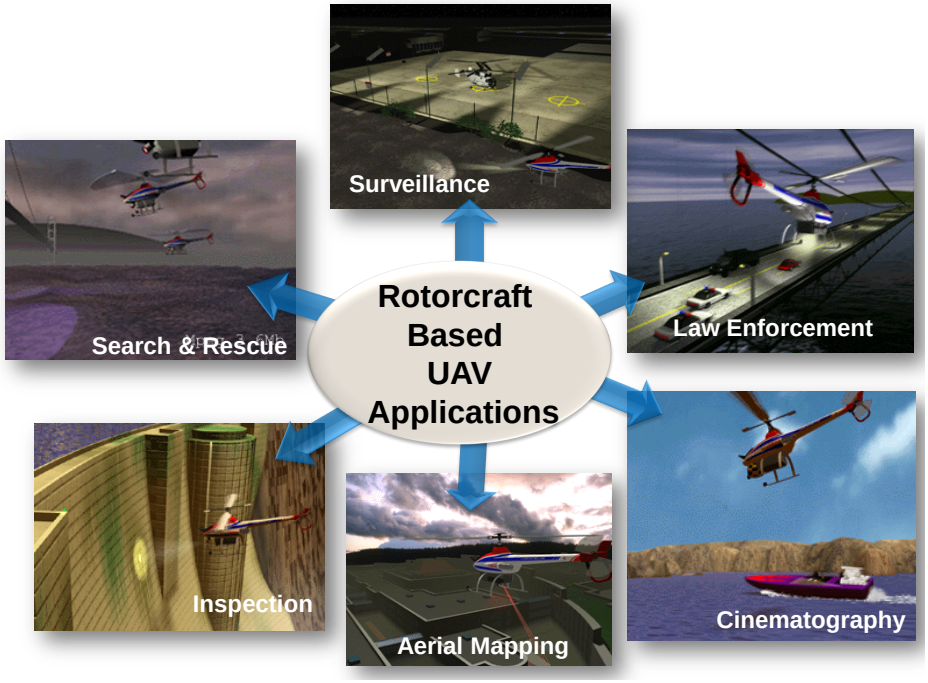


Figure 1.1 Examples of typical applications of helicopter based UAS.

single rotor helicopter configuration offers numerous potential capabilities as shown in Figure 1.1. Potential applications include surveillance, aerial mapping, cinematography, high structure inspection and monitoring operations. Furthermore, the helicopters are also known to be indispensable air vehicles for finding and rescuing stranded individuals or transporting accident victims. Police departments use them for traffic control, ground support, high-speed car pursuits, observation, air patrol and control of large-scale public events or public order incidents. Fire fighters use helicopters for precise delivery of fire extinguishing chemicals to forest fires. More and more electric power companies are using helicopters to inspect towers and transmission lines for corrosion and other defects and to subsequently make repairs.

All of these applications require low altitude flight capability to hover over targets and demand dangerous close proximity flight patterns. Any fatal error presents immense risks to human pilot safety. Since the rotorcraft based UAS possesses unique manoeuvre capabilities to hover and cruising at a lower speed, rotorcraft UAS are deemed more suitable for these kinds of application compared to fixed wing UAS. An unmanned helicopter system with built-in autonomous capability can eliminate unnecessary risks

to the human pilot and will increase the effectiveness of the helicopter operation. Furthermore, the helicopter based UAS is also capable to take-off and land vertically from limited space, hence making it much more efficient than fixed wing UAS that usually requires a runway for take-off and landing.

Automation of these specific missions requires development of an automatic flight control system (AFCS) for helicopter based UAS. The automatic control function is responsible for manipulating the inputs to a dynamical system to obtain a desired effect on its outputs without human intervention in the control loop. This is achieved using computation of control laws that calculate input commands for vehicle actuators (i.e. rotor cyclic input, aileron, elevator etc.) to produce torques and forces acting on the vehicle in controlling its 6 degrees of freedom (DOF) motion (position, orientation, and their time derivatives). The work presented in this thesis will focus on the development of a flight control system for a miniature helicopter UAS in hovering mode, which is necessary for aerial surveillance and monitoring operations.

1.2 PROBLEM STATEMENT

A miniature single rotor UAS is regarded as an inherently unstable non-linear system with fast responsive dynamics due to their small size. The research communities in industries and universities use commercially available model scaled helicopters as experimental and development airframes for Guidance, Navigation and Control (GNC) research due to airframes' sufficient payload capability. Due to the helicopter's unstable dynamics nature, low level controls such as velocity and attitude feedback controllers are required to stabilise the aircraft's 3 DOF motion. If the helicopter UAS fails to receive stabilising control commands for even a brief period, it will most likely become unstable and crash. This makes conducting research on a helicopter platform challenging where there is no room for error or the consequence can be disastrous.

The AFCS for helicopter based UAS was traditionally designed using the linearisation principle of rigid body equation of motion at various conditions throughout the flight envelope [Mettler, 2003, Kendoul, 2012]. Subsequently, the controller system

is then designed for each different flight condition such as hovering flight or forward flight at different velocities. Since we have multiple controllers that provide satisfactory control for different operating points, gain scheduling approach is used to determine the current flight operating region and to activate the appropriate linear controller. Several measured variables from the on-board instrumentation such as airspeed, dynamic pressure or altitude can be used to trigger a specific linear controller relative to the current flight operating region. However, such control technique suffers performance degradation when performing large amplitude manoeuvres [Mettler, 2003, Kendoul, 2012, Valavanis, 2007]. Furthermore, a scheduling control method can results in a control signal that jumps to different values at each model transition. This effect needs to be reduced in practice as the sudden increase in the control action can result in dangerous sudden movement of the helicopter [Joelianto et al., 2011]. In addition to limitation of linear approaches, the control problem becomes much more challenging to solve as the operation of the helicopter also exposes them to varying atmospheric disturbance.

Numerous advanced non-linear controllers such as feedback linearisation, adaptive control and non-linear model based approaches have been suggested in the literature to overcome the limitation of linear approaches with successful implementation in real flight tests [Kendoul, 2012, Cai et al., 2010]. Kendoul [2012] further argues that even though numerous advanced non-linear control approaches were proposed to improve the AFCS performance, significant improvement in flying capabilities has not been achieved with the non-linear controllers when compared with standard linear controllers. This is due to the fact that the linear controllers such as the widely used PID, LQR and H_∞ methods are robust and mature enough for the helicopter based control application. However, Kendoul [2012] in his review proposed that certain aspects of AFCS development can be further investigated and improved. The lists of AFCS developments that are required for further improvement are given as follows:

- The development of a general purpose, flexible and self-tunable flight controller that can be deployed into different UAV airframes in a shorter development time. Such a controller design should include an adaptation mechanism to the internal

model after physical changes occur (new installation of sensors or additional payload).

- The development of a robust flight controller that can perform well in strong disturbances such as windy or severe weather conditions.

- The development of a reconfigurable flight controller that redefines the control strategies based on flight modes, mission conditions and fault scenario.

The capability of the helicopter based UAS flight controller can be further improved with the development of self-tunable and flexible controllers based on the neural network approach. The learning based method using neural network is well known to be a universal approximator, and hence is able to map complex input-output relationships using the test data [Hornik et al., 1989]. This unique ability of the neural network approach provides a good basis for development of a flexible and self-tunable flight controller for different rotorcraft platforms. Besides the ability to learn complex mapping, the neural network based controller can further be equipped with adaptive capability to parametric uncertainty and unknown flight dynamics. This leads to better controller performance under varying flight operating conditions since the adaptation capabilities in model estimation should give better accuracy of estimation of the non-linear process.

The majority of the development process of helicopter systems has evolved around the dynamics modelling and control system design which contributes around 25-50% of total development work [Padfield, 2007]. The rotorcraft dynamic should be sufficiently modelled in various operating conditions to prevent major control performance degradation resulting from the poor predictive capability of the mathematical model developed. Padfield [2007] points out that poor model prediction can result in redesign efforts to improve or fix problems arising from poor flight performances and flying qualities. Subsequently, this leads to increased usage of resources, time and cost due to redesign efforts.

It is an undeniable fact that the helicopter is a complex dynamic system and the modelling task requires a large amount of effort and extensive modelling skills. The first principle modelling method based on Newton laws are commonly used to infer

the dynamics of the helicopter. However, this method requires a significant amount of work to estimate model parameters through physical measurements and experiments [Mettler, 2003]. The method also demands solid theoretical knowledge and experiences about the rotorcraft flight and has the potential to produce unreliable results unless performed with extreme care. After obtaining the non-linear model, the unmanned helicopter dynamics is described using linearised model at various flight conditions such as hover and forward flight conditions. However, the linearised dynamic models are only valid within the range of operating conditions and multiple linear models are needed to extend the flight operating condition outside the linear range [Mettler et al., 2002b, Kendoul, 2012].

Simplified mathematical models have been developed over the years for helicopter UAS flight controller design [Shim, 2000, Bisgaard, 2007, Heffley and Mnich, 1988]. However, these models suffer performance degradation due to unmodelled or hard to model dynamic effects that are not incorporated in the mathematical model itself. There are numerous effects that are typically omitted from the model such as the ground effect, servo dynamics, rotor speed variation, sensor lag or actuator kinematic non-linearities. Simplifying assumptions or omitting the mentioned dynamic effects can introduce significant errors from the model prediction [Garratt and Anavatti, 2012]. Therefore, a more comprehensive modelling approach is required for the modelling of the dynamics of the helicopter UAS which fully exploit the capabilities presented by their complex dynamic behaviour. The neural network approach again can be used as an alternative method in helicopter dynamic modelling. Fast and simpler development of the dynamic model using the neural network approach should reduce the cost associated with the development of large aerodynamic databases [Calise and Rysdyk, 1998].

The neural network approach had been used for mathematical modelling or control application with success [Paliwal and Kumar, 2009, Calise and Rysdyk, 1998]. However, the neural network modelling method has several disadvantages such as high computational resources required for training, slow convergence rate, being prone to over-fitting [Tu, 1996, Wilamowski, 2011a, Norgaard, 2000]. Furthermore, neural networks are also known to be a ‘black box’ model and have limited capability to express a causal

relationship between inputs and outputs. Several recommendations have been made in this thesis in terms of model structure selection, validation and training methods to overcome the problems presented by the neural network based modelling. Better selection of neural network model structure should improve the prediction of the model while using more advanced neural network architectures other than standard multi-layered perceptron should reduce the prediction model training time. This should be beneficial to the real-time implementation of the adaptive flight control system. The task to explain how the neural network solves the modelling problems is outside the scope of this thesis. Nevertheless, recent research has emerged in NN literature to extract regression rules that explain knowledge gained by the NN model [Setiono et al., 2002, Jianguo et al., 2011, Kamruzzaman and Islam, 2010].

The ability to model the time varying dynamics of a UAS helicopter is also important in the development of adaptive type flight controller. Typically, a neural network model derived from the off-line based training (batch training) will not be able to represent all the operating points of the flight envelope very well [Samal, 2009, Ljung and Soderstrom, 1983]. Several attempts have been made in previous studies to update the neural network prediction model during flight using mini-batch off-line training on a smaller number of data samples [Samal, 2009, Samal et al., 2008, 2009, Puttige, 2009, Puttige and Anavatti, 2006]. However, the proposed method can only be employed to reasonably small networks and is limited to model uncoupled helicopter dynamics due to high computation cost. In order to accommodate the time-varying properties of helicopter dynamics which change frequently during flight, a recursive based learning algorithm is required to properly track the dynamics of the system under consideration. The application of a recursive type neural network should further improve the prediction and adaptability of the dynamic model.

This thesis attempts to overcome the problem by presenting neural network based modelling and control frameworks that greatly reduce the development time, cost and resources needed to design a high performance control system for helicopter based UAS. The neural network based approach to the system identification of unmanned helicopter dynamics has shown promising capability to facilitate the development of accurate flight

models [Samal et al., 2009, 2010]. Furthermore, the parallel nature and fast adaptability of neural networks are well suited for adaptive control design and implementation for unmanned helicopter systems.

1.3 RESEARCH OBJECTIVES

The goal of this research project is to design an automatic flight control system (AFCS) for autonomous hovering of a single rotor helicopter UAS platform. The performance and effectiveness of the proposed adaptive controller is evaluated in hovering flight tests. The objectives set forth for the research work are listed as follows:

1. Develop the system identification algorithms for modelling the non-linear dynamics of the helicopter UAS in flight.

2. Develop a suitable controller using the identification algorithm developed for controlling the hovering manoeuvre.

3. Integrate the helicopter platform with the necessary avionics and experimental apparatus, test and validate the system identification and flight control methods.

1.4 THESIS CONTRIBUTIONS

The major contributions of the thesis are listed as follows:

1. Identification of helicopter UAS dynamics using NN based system identification methods.

   In this study, the neural network based system identification algorithms are developed to model the non-linear dynamics of the UAS helicopter under consideration from the flight test data. Since the helicopter dynamics are non-linear, the neural network system identification approach using NNARX (Neural Network-Auto Regressive structure with eXtra inputs) model structure is used to address such a problem. The NNARX structure is able to infer complex non-linear relationship between inputs-outputs data sets and demonstrate the ability to adapt to changes

in an operating condition. Previous research work on system identification of unmanned helicopter system usually neglect proper selection of model structure and follow a trial and error approach which leads to improper selection of neural network structure. In order to optimise the network structure selection that lead to better generalisation and prediction quality, the neural network model structure in this study was carefully selected using the proposed Lipschitz coefficient and model validity tests method.

Three types of NN architectures are used to model the dynamics of the helicopter; namely the MLP, HMLP and the modified Elman network. The HMLP and modified Elman network are proposed in our work to reduce the total number of weights used in the MLP network. If the NN training is conducted using a recursive type training method, the reduced number of NN weights should reduce the amount of computation needed to train the NN model, which makes real-time system identification possible. Although the total number of weights for both HMLP and modified Elman network are lower than the MLP network, the prediction performance of both models are on par with the prediction quality of MLP network.

2. The development of recursive type system identification algorithm for on-line modelling of helicopter dynamics.

   The predicted response from the off-line neural network model is found suitable for modelling the UAS helicopter dynamics correctly. However, the model identified through the off-line modelling has several drawbacks. The approach has difficulties in representing the entire flight operation very well because of the time varying nature of helicopter flight dynamics. The recursive type training such as the recursive Gauss-Newton (rGN) method is adopted in this study to overcome such a problem. The recursive method presented in this work is suitable to model the UAS helicopter in real time within the control sampling time and computational resource constraints. Satisfactory prediction quality is achieved with the rGN training method even with incorrect model structure assignment. The generalisation and adaptability performance of the model can be further

improved by proper network structure selection. This can be obtained with the aid of the k-fold cross validation method. Furthermore, the implementation of the proposed network architectures, namely the HMLP and modified Elman networks, is found to improve the learning rate of NN prediction; and this facilitates the real-time implementation of the NN based system identification.

3. The development of real-time Neural Network based Approximate Predictive Control (APC) scheme with constraints.

   The Neural Network based Approximate Predictive Controller (NNAPC) is designed and developed using the NN model identified either from the off-line or on-line system identification algorithms. The complexity of the proposed controller algorithm can be handled well enough by the embedded system even with the implementation of the on-line system identification algorithm. The control method is proposed to overcome the demanding computation efforts of non-linear Model Predictive Control (NMPC)'s optimisation procedure. The main difference between NNAPC with linear MPC and other NMPC methods lies in the prediction operation of NNAPC. Instead of relying on the prediction from single linear or non-linear model, the NNAPC controller extracts linear models from the identified NN model at each sampling time and uses it to predict the future plant response within the specified time horizon. The proposed NNAPC algorithm is able to achieve satisfactory tracking control performance with different degrees of control autonomy. The NNAPC controller is fine tuned for each of the control channels (roll, pitch, yaw and altitude) and the selected tuning parameters are then used for the testing of full autonomous hovering control. The NNAPC controller approach is also tested and proved to be robust enough to handle variation in helicopter dynamics and demonstrate good disturbance rejection performance to input disturbances. The flight test results indicate the efficiency of the NNAPC controller to achieve autonomous hovering.

4. The development of a 6 DOF test rig for safe control flight test of helicopter UAS.

   Attempting to control a remote control helicopter is difficult and careful experi-

mentation is essential in building a working UAS helicopter prototype. For the purpose of testing the AFCS developed in this research project, a six degree of freedom (DOF) test-rig is developed to ensure safe flight test of the helicopter. The test rig is designed and built mainly as a safety device for preventing crashes and out-of-control flight. Several test stand design examples for helicopter control tests are available in the literature, but most of these stands are only limited to fewer DOFs that cannot be used to simulate free flight of the UAS [Cetto et al., 2009, Amidi, 1996, Kim and Tilbury, 2004, Vitzilaios and Tsourveloudis, 2009, Vilchis et al., 2003].

The proposed design in this work extends the limitation of the current test rig design which allows the execution of a full 6 DOF movement with limited work space. The safety test rig system is equipped with instrumentation sensors that provide position and orientation measurements of the helicopter UAS. Such a system can provide the ability to test the controller continuously regardless of the weather conditions. Subsequently, the development of such device reduces dependency for an experienced helicopter pilots while conducting flight tests.

1.5 THESIS ORGANISATION

This thesis is organised into seven chapters. The first chapter introduces the motivation, research objectives and contribution of this research. The remaining chapters are organised as follows:

Chapter 2 presents a review on the development, application and classification of the rotorcraft UAS types. Different approaches and methods in helicopter dynamics modelling and system identification are explored. Next, literature review on neural network based system identification is presented. The selection of neural network model structure and training algorithms employed in the literature are given special emphasis. Then, the existing neural network based adaptive control design techniques are discussed. Published research findings that have influenced this thesis work are explicitly highlighted.

Chapter 3 presents the overview of the helicopter platform and avionic systems used for the development of the autonomous helicopter UAS. For flight control testing purposes, a safety test rig has been developed to constraint the helicopter movement and avoiding fatal crashes which would arise from possible hardware failures or programming errors.

Chapter 4 presents the procedures of the off-line system identification of helicopter dynamics using the neural network approach. Then, the on-line (recursive) training method is proposed to improve the adaptability of the predicted model. The prediction of the neural network model employed in this study is further improved using Hybrid Multi-Layered Perceptron (HMLP) and modified Elman networks which significantly reduce the number of weights in the network and reduce the required computation time.

Chapter 5 presents the model selection and validation results from the proposed neural network based system identification methods. The optimal network structures for system identification are found using the proposed validation method used in the previous chapter. The performance of standard Multi-Layered Perceptron (MLP) is then compared to the HMLP and modified Elman network performance.

Chapter 6 presents the model predictive control methodologies using the dynamic model identified using the neural network based system identification approach. It proposes solutions to the drawbacks of neural network based MPC real-time implementation by introducing the application of Neural Network based Approximate Predictive Control (NNAPC). The prediction of NNAPC is provided using linear models extracted from the identified NN model obtained from either off-line or on-line modelling techniques.

Chapter 7 presents the flight tests implementation and validation results of the proposed adaptive flight controller. In this chapter, the tuning procedures of the NNAPC flight controller are presented. The proposed flight controller is designed in several development stages where the best tuning parameters obtained from

different stages are then used as the basic tuning values for hovering flight controller. Subsequently, the flight test results from the tuning procedures are then presented, including those from the dual channel control (pitch-roll), triple channel control (pitch-roll-yaw), altitude control and hovering control. Then, the proposed NNAPC flight controller performance is tested and validated under conditions such as dynamic parameter variations and input disturbances.

Chapter 8 presents the conclusions about the research project and recommendations for future work.

Chapter 2

LITERATURE REVIEW

The purpose of this chapter is to present a literature review on helicopter based UAS technology, helicopter dynamics modelling, system identification approaches and the development of neural network (NN) based flight controllers for autonomous flight of an unmanned helicopter system.

The chapter is organised as follows. In Section 2.1, the potential areas of application, the unique features and classification of helicopter based UAS are presented. The components in autonomy enabling function that are required to achieve autonomous operation are also described along with the challenges involved with the design of automatic flight control system (AFCS). Flight vehicle system identification and first principle modelling are reviewed in Section 2.2. NN based system identification and its fundamental aspects such as model structure selection and training algorithm are discussed in detail in Section 2.3. Different control techniques, such as classical and NN based techniques used for flight vehicle control are provided in Section 2.4, together with a brief review on model predictive controller. Section 2.5 provides a brief summary of the chapter.

2.1 INTRODUCTION

An Unmanned Aerial Vehicle (UAV) can be defined as an aircraft designed with no pilot on-board which can performs various roles of piloted aircraft. A multi physical system that includes the UAV, ground control station (GCS), payload and communication architecture is defined as Unmanned Aerial System (UAS) with no human elements

on-board in any of the system components [Roadmap, 2005]. According to US Defence Department UAS Roadmap 2010-2035, humans are not a part of UAS but rather an external system interacting with the UAS [Roadmap, 2005]. There are a number of important fields of science and technology that are directly related to UAV research such as aerodynamics, propulsion, structural, flight dynamic and control, flight performance and electronic system integration into UAV platform. On the other hand, a fully autonomous UAV indicates that an airframe is capable of performing given missions successfully within a pre-defined scope. This can be achieved through the use of on-board sensors and manipulation systems without human or external system involvement [Kendoul, 2012].

Rotorcraft or helicopters have been used as an experimental platform for UAV research throughout academics and aeronautical industries. Helicopter based UAS offers many potential capabilities in both civil and military applications such as surveillance, aerial mapping, cinematography, high structure inspection, natural disaster damage assessment and monitoring operations. Technological advancements in this exciting area have enabled a helicopter based UAS to operate autonomously, which reduces risk and a pilot's workload while flying in close proximity flight patterns.

Any type of aircraft may serve as the base airframe for a UAS application. Traditionally, fixed-wing aircrafts have been preferred as the aerial platforms simply because of their simple structures, and being efficient and easy to build and maintain. The autopilot design is easier for fixed-wing aircraft than for rotary-wing aircraft because the fixed-wing aircraft have relatively simple, symmetric, and decoupled dynamics [Shim, 2000]. However, helicopter-based UAS have been desirable for certain applications where the unique helicopter's flying capabilities are required. The rotorcraft can take off and land within limited space and they can also hover and cruise at very low speed which makes them the perfect vehicle for tracking or searching out ground targets. The helicopter based aerial platform is also much more efficient than fixed wing type aircraft that usually requires a runway for take-off and landing. However, the helicopter based UAV capabilities are also limited with disadvantages which are listed as follows:

- More complicated mechanical structure.

- Inefficient flight dynamics: lower maximum speed, shorter mission range.
- More accurate and complicated navigation sensor requirement.
- Inherently unstable and relatively poorly known dynamics. Difficult control system design.

Typical helicopters or rotorcraft based UAS are classified into five categories that differ on several characteristics [Kendoul, 2012, Eisenbeiss, 2004]. Figure 2.1 shows the classification of helicopter UAS based on attributes such as the total weight of the aircraft, payload, range, service ceiling and endurance. Most unmanned aerial vehicle platforms available at the University of Canterbury consist of rotorcraft based UAS from Category 3 and 4. The UAS from Category 3 is based on commercially available remote control (RC) helicopter and requires considerable amount of work to integrate the autopilot system into the aerial platform. Nevertheless, the aerial platform used in this research was selected from Category 3 UAS which offers larger payload capacity and longer endurance than Category 4 or 5. Thus, better payload capacity and endurance would make the selection of Category 3 UAS much more suitable for longer monitoring and surveillance applications such as aerial mapping, cinematography, high structure inspection and natural disaster damage assessment.

A helicopter based UAS requires three main autonomy elements such as guidance, navigation and control (GNC) to enable them to execute above-mentioned missions autonomously. The overview of autonomous system components for helicopter based UAS is given in Figure 2.2. As described in Kendoul [2012], the navigation function for helicopter UAS involves the process of acquisition and analysis of sensory data and how to use them to infer the vehicle's state informations (position, orientation, velocity). The navigation process also includes the capabilities to build an internal model of the surrounding environments within UAS operation using perception techniques such as mapping, object recognition, obstacle and target detection. These capabilities are essential for operating a UAS to ensure that it can successfully complete an assigned mission. The guidance function for a helicopter based UAS demonstrate the high level planning and decisions making process to enable successful missions or goal execution.

Figure 2.1 The categories of helicopter based UAS based on size, payload and flight performance.

Generally, a guidance system uses information from the navigation blocks and mission goals to make appropriate decisions to generate trajectory reference for the low level controller.

The automatic flight control system (AFCS) is a key autonomy-enabling functionality that increases the helicopter based UAS autonomy level from 'level 0' to 'level 1' on the autonomy level scale defined in Kendoul [2012]. In Level 0 (Remote Control), the control commands to the UAV are given by a remote external system or pilots, whereas in Level 1 (AFCS), the control commands to the unmanned vehicle are computed by the flight control system to control the helicopter's position and orientation according to the given references. For the current project, the automatic flight control system (AFCS) is the main focus of our research since the AFCS is the first key autonomy

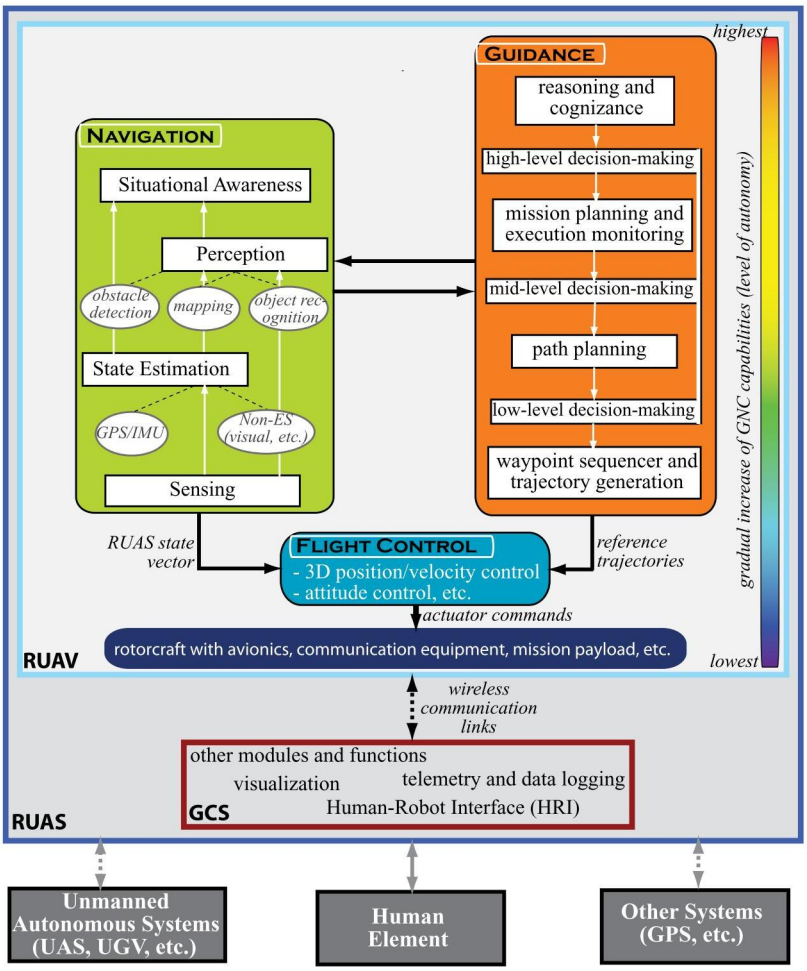


Figure 2.2 The structure overview of guidance, navigation and control system for helicopter based UAS [Kendoul, 2012].

element that enables a UAS to be autonomous.

2.2 HELICOPTER DYNAMICS MODELLING AND SYSTEM IDENTIFICATION

Numerous advanced control designs such as non-linear control and adaptive control have been developed over the years to overcome the limitation of linear control. These designs typically require a mathematical model representation of the system to be controlled in the form of differential equations. There are two basic ways to obtain a mathematical model of the system. The first method is to deduce the model in a constructive manner using first principle modelling (direct use of Newtonian Laws to describe the system behaviour). The second method is to infer the model from an experimental data set collected during experiments with the system. The latter approach of deriving the dynamic model of the system is also known as system identification approach.

The first approach of modelling involves a comprehensive mathematical description about the dynamic system. The system itself is divided into multiple components that contribute to the total forces and moments affecting the system. Various unknown parameters in the mathematical model need to be approximated or measured through the mean of experimentation or measurement of physical parameters of the system. Thus, this would make the modelling task more complex and time consuming [Norgaard, 2000]. On the other hand, the system identification approach offers a more practical and simple solution to obtain a mathematical model of the system with a reasonable effort. The system identification approach can also be viewed as a function fitting process to obtain a model that best describes the measurement data from the experiment. However, the main drawback of this approach is that the experiment needs to be conducted in such a way that the system under consideration needs to be excited through its entire range of operation [Norgaard, 2000].

The UAS helicopter dynamics is known to be a non-linear and time varying model of high order multiple-input multiple output (MIMO) system. It is also an under-actuated system with four actuator inputs: main rotor collective pitch, lateral cyclic pitch, longitudinal cyclic pitch and tail rotor collective pitch. The first principle modelling approach uses physical principles and Newton's second law to describe the dynamic

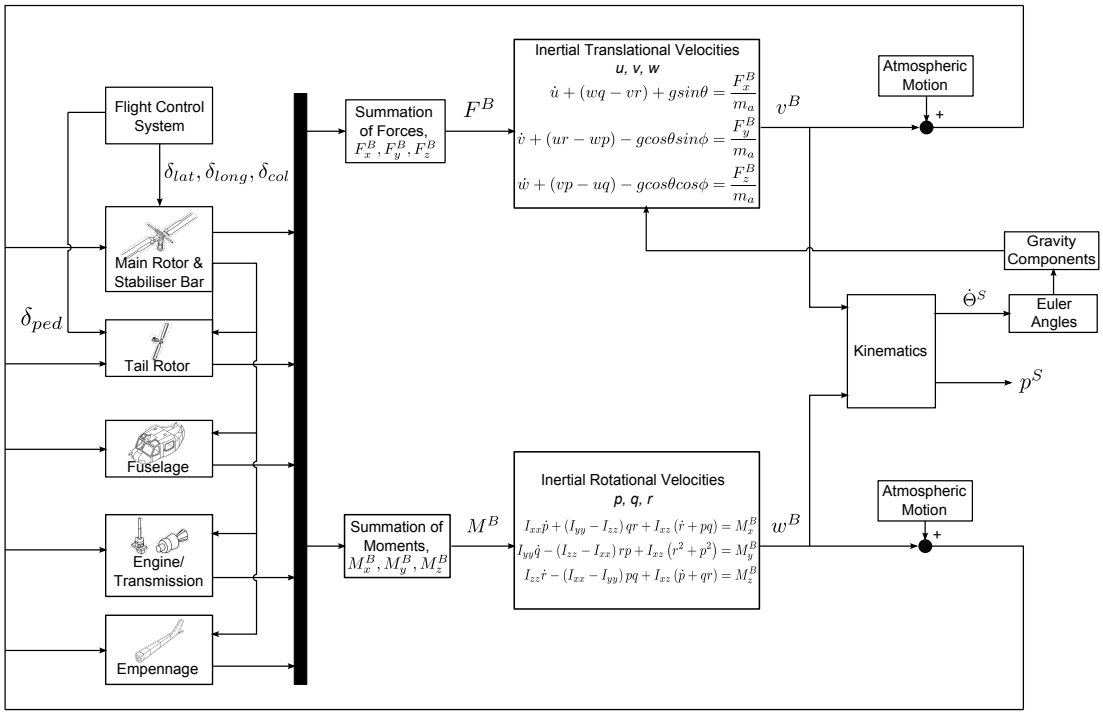


Figure 2.3 Typical arrangement of component forces and moments generation in helicopter simulation model. Figure adapted from Padfield [2007] and Cai et al. [2013].

system behaviour. This procedure is also known as a lumped parameter approach where all components are described individually and then collected as a complete system [Padfield, 2007, Shim, 2000]. The helicopter dynamics is mainly considered as a composition of the body mass, the main rotor and the stabiliser bar, the tail rotor and the active yaw damping system. Figure 2.3 illustrates the typical arrangement of component forces and moments generation in a helicopter simulation model. The aerodynamic drag effects from the fuselage, horizontal and vertical stabiliser are normally neglected as they play a less important role under hovering and slow flight conditions for model scaled helicopter.

The helicopter is basically considered as a rigid body on which forces, $F^B = \begin{bmatrix} F_x^B & F_y^B & F_z^B \end{bmatrix}^T$ and moments, $M^B = \begin{bmatrix} M_x^B & M_y^B & M_z^B \end{bmatrix}^T$ are exerted. Its motion is described using Newton-Euler equations of motion expressed in the Body Reference Frame [Shim, 2000]. Let superscript S and B represent spatial (inertial) and body reference frame respectively. The twelve dimensional state vectors consist of the helicopter's centre of gravity (CG) position vectors in spatial reference frame, $p^S = \begin{bmatrix} p_x^S & p_y^S & p_z^S \end{bmatrix}^T \in \mathbb{R}^3$; Euler angles (roll, pitch and yaw), $\Theta^B = \begin{bmatrix} \phi^B & \theta^B & \psi^B \end{bmatrix}^T \in \mathbb{R}^3$

in body reference frame; helicopter linear velocity in body reference frame, $V^B = \begin{bmatrix} u^B & v^B & w^B \end{bmatrix}^T \in \mathbb{R}^3$ and helicopter angular velocities in body reference frame, $w^B = \begin{bmatrix} p^B & q^B & r^B \end{bmatrix}^T \in \mathbb{R}^3$. The overall dynamic system of a helicopter can be divided into kinematics (2.1) and system dynamics (2.2) as follows:

$$\begin{aligned} \dot{p}^S &= R^{B \to S} V^B \\ \dot{\Theta}^S &= \Psi(\Theta)\, w^B \\ \Psi(\Theta) &= \begin{bmatrix} 1 & 0 & \tan\theta\cos\phi \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \end{aligned} \tag{2.1}$$

$$\begin{aligned} \dot{V}^B &= \frac{1}{m} F^B - w^B \times V^B \\ \dot{w}^B &= J^{-1}\left(M^B - \left(w^B \times J w^B\right)\right) \end{aligned} \tag{2.2}$$

where m is the total mass of the vehicle, J is vehicle inertia matrix, $R^{B \to S}$ is rotational matrix from body reference to spatial reference frame.

The forces F^B stated above are the total sum of aerodynamics forces generated from the main rotor, tail rotor, fuselage, vertical and horizontal stabilisers and gravitational force. The total moments M^B are generated by aerodynamic forces from various vehicle components and moment generated by main rotor gyroscopic effects [Gavrilets et al., 2003, Bisgaard, 2007]. Basically, the aerodynamic forces and moments generated from main rotor and tail rotor are non-linear functions which depend on the operating conditions, vehicle motion characteristics and control inputs [Kendoul, 2012]. The control inputs associated with pilot commands are defined as, $\delta = [\delta_{lon} \quad \delta_{lat} \quad \delta_{col} \quad \delta_{ped}]^T \in \mathbb{R}^4$ where δ_{col} and δ_{ped} are collective pitch of main rotor and tail rotor respectively, δ_{lon} and δ_{lat} are longitudinal and lateral cyclic pitch respectively which control the inclination of main rotor's tip path plane according to their respective directions. The illustration of the state and input variables that features in the helicopter flight dynamics is given in Figure 2.4. The control input commands are normalised between -1 to 1 for lateral cyclic, longitudinal cyclic and tail rotor's collective pitch, while the main rotor's collective pitch command is normalised between 0 to 1 range.

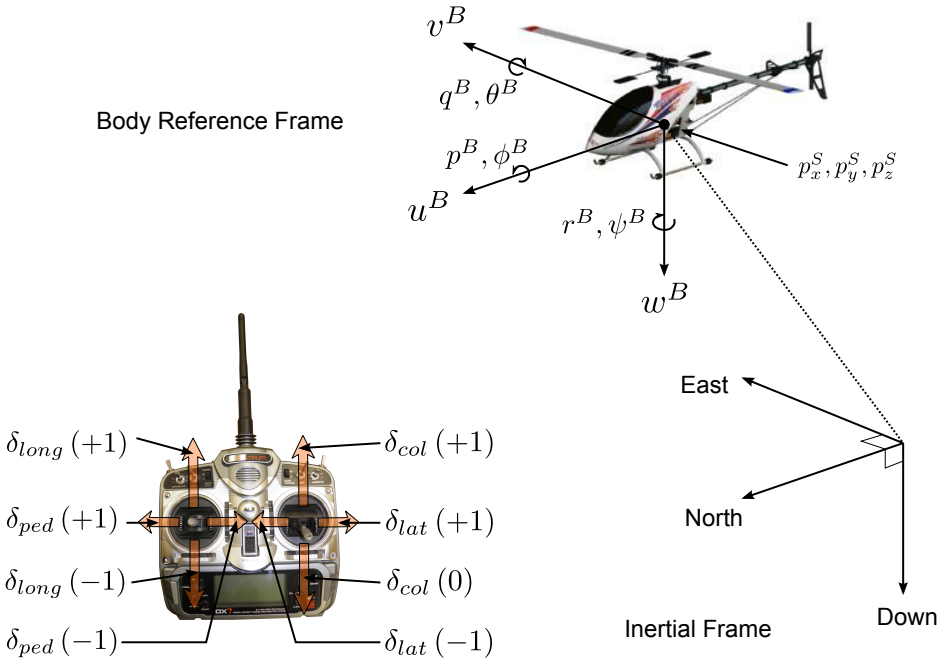


Figure 2.4 The representation of state and input variables.

Kendoul [2012] has suggested that in order to obtain more accurate helicopter dynamics model, the basic rigid body model needs to be extended with helicopter's rotor dynamics and detailed aerodynamic model using theories from simple momentum theory or blade element theory. Detailed formulation of large scaled helicopter's non-linear dynamics model can be found in Prouty [1986], Padfield [2007], Bramwell et al. [2001], Johnson [1980]. For a smaller scale helicopter, other dynamics from the actuator and stabiliser bar also need to be included in the rigid body model to increase the model fidelity [Mettler, 2003]. The dynamics model presented above depends on many parameters which need to be carefully identified through direct measurement of geometrical or structural data. For aerodynamic parameters estimation, elaborate experiment needs to be performed using wind tunnel facilities. It requires a considerable amount of theoretical knowledge and experiences about rotorcraft flight and potentially would not produce highly accurate results unless performed with extreme care. In some cases, the agreement between the predicted and measured dynamic behaviours is unsatisfactory because of the accumulated uncertainty and modelling simplification [Mettler, 2003, Kendoul, 2012].

Apart from high fidelity modelling theories from standard full scaled helicopter, several minimum complexity mathematical models can also be used. Several good examples of the development of minimum complexity mathematical model are given in Shim [2000], Bisgaard [2007] and Heffley and Mnich [1988]. Although such approaches can be employed to build a simulation model, the highly non-linear aerodynamics interaction between body components and the behaviour of the high order dynamics of a rotorcraft is usually hard to model using the first principle approach (direct physical understanding of forces and moments balance of the vehicle) and such an approach can be inaccurate [Mettler, 2003, Budiyono et al., 2009, Deng et al., 2011]. A significant effort is necessary to validate the theoretical model and re-evaluate any potential shortcomings associated with the model.

Most of the works adopting the first principle modelling approach in helicopter modelling use the developed model for control design without detailed validation against flight data [Kendoul, 2012, Bisgaard, 2007]. Among the few successful works that use first principle modelling to address the problem of physical parameter estimation with experimental validation are Nonami et al. [2010] and Gavrilets et al. [2001]. In Nonami et al. [2010], the non-linear helicopter dynamics is linearised to obtain two linear state equations that describe the helicopter's lateral and longitudinal motion. The linear analytical model includes the rigid-body dynamics, main rotor dynamics and aerodynamics, stabiliser bar dynamics and aerodynamics effect from other main body components. The parameters of the model are determined using direct physical measurements and manufacturer specifications for different helicopter platforms. For parameters that are more difficult to obtain such as moment inertia in different axes, the values was measured roughly and manually retuned to match the flight data. The linear models have been validated for three different helicopter UAV platforms and results show good fit with the flight data for operating condition around hovering flight.

Gavrilets et al. [2001] describes 17 states of the non-linear dynamics model of an acrobatic helicopter UAV which includes states for longitudinal and lateral main rotor flapping, the rotor speed and an integral of the rotor-speed tracking error in addition to rigid body dynamics. Flight test experiments were used to estimate several key

parameters, such as the equivalent stiffness in the rotor hub and equivalent fuselage frontal drag area. The model's accuracy was verified using comparison between model predicted responses and responses collected during flight test. Even though the prediction results obtained from the non-linear model are adequate for a variety of flight conditions, again, extensive validation needs to be carried out to fine tune many parameters in the model.

Since the helicopter is a highly non-linear multi-variable system with some degree of coupling effect in its dynamics, it is preferable to design a controller that includes the effect of coupling between various inputs of the helicopter. However, it is not always a practical approach to include complete coupling effect in the controller design as this will result in increasing computational complexity and resources. Decoupling of the coupled dynamics into a much simpler dynamics representation with separated actuators not only decreases the computation burden but also serves as an effective way to incrementally design the controller for a complete dynamic system.

In order to simplify the modelling problem, the dynamic model of a helicopter was described and partitioned into smaller identification problems such as coupled roll-pitch dynamics, heave dynamics, yaw dynamics or coupling of heave and yaw dynamics or with some coupling combination among these coupling cases [Mettler, 2003, Tischler and Remple, 2006]. Different degrees of coupling exist between dynamic channels when considering the helicopter as a MIMO system. As an example, in the longitudinal channel, the relationship of longitudinal cyclic pitch and longitudinal angular velocity is the main feature of the longitudinal channel. Other coupling effects that include lateral cyclic pitch, collective pitch and tail rotor's collective pitch have some effect on the longitudinal angular velocity in a certain frequency range. Castillo-Effen et al. [2007] propose calculation methods to quantify the degree of interaction between dynamic channels by using the relative gain array number and the diagonal dominance function that quantify decoupling. By using the proposed calculation methods, the helicopter dynamics were found to behave strongly as two sets of a Two Inputs-Two Outputs (TITO) system. Strong coupling exists between lateral and longitudinal channels ($\phi, \theta, \delta_{lat}$ and δ_{long} pairing), as well as mild coupling between collective and pedal channels (r, w, δ_{ped}

and δ_{col} pairing). The coupling between collective and pedal channels can be further simplified into two sets of a Single Input-Single Output (SISO) system similar to the assumption used in Mettler [2003] and Samal [2009].

In system identification approach, the non-linear mathematical model from the first principle approach can be identified from the experimental input-output data using error minimisation or optimisation methods. In Kim and Tilbury [2004], a system identification of a model scaled helicopter using time domain analysis tool was presented. The interaction between flybar and the main rotor blade was included in the model development which considers the effects of flybar flapping mechanism on helicopter stability. The identification of the decoupled SISO transfer function was obtained using the least square error minimisation between the time domain response predicted by the transfer function and the measured experiment data. The identification experiment was done in special test benches that restricted the motion of the helicopter to one degree of freedom (DOF). However, results from the time domain system identification indicated that the prediction obtained from the model gave a fairly poor prediction performance and did not track the faster dynamics of the system [Mettler, 2003].

One of the most popular and effective method to identify the dynamics model of small scaled helicopter UAV was based on the identification method proposed by Mettler et al. [2002b]. The method was based on the frequency response identification technique that used analytical software package such as Comprehensive Identification from Frequency Responses (CIFER®) software. This method was mainly used for military fixed wing and rotary wing aircraft system identification [Tischler and Remple, 2006]. Mettler et al. [2002b] carried out a detailed identification work from the collection of flight data and sufficiently estimate linear models in hovering and cruise flight conditions for the Yamaha R-50 helicopter UAV. Both flight conditions were accurately described by the identified linear models with additional dynamic effects such as the first-order rotor and stabiliser bar dynamics with no inflow dynamics effect. The linear models accurately captured the vehicle dynamics roughly around the nominal operating points. Even with the acceptable identification result, Mettler [2003] has further suggested to identify more linear models that represent adequately a large portion of the flight envelope

with sufficient accuracy for the control design. This further motivated us to find more comprehensive modelling solutions that cover the extended operating conditions outside the linear range much more effectively.

2.3 NEURAL NETWORK BASED SYSTEM IDENTIFICATION

Although various simplified mathematical models have been developed over the years for designing the flight controller for helicopter based UAS, many models suffer performance degradation due to many unmodelled dynamics that are not incorporated in the mathematical model itself. Consider the altitude control channel, significant errors can arise due to simplifying assumptions or omitting several factors such as servo dynamics, rotor speed variation, sensor lag, ground effect, actuator kinematic non-linearities and rotor inflow lag associated with the rate of change of the blade pitch [Garratt and Anavatti, 2012]. These dynamic effects are hard to model exactly, but can be compensated by the use of learning methods such as fuzzy inference system or artificial neural network (ANN) approach. The fuzzy inference system is a modelling approach that represents the input and output mapping through the use of fuzzy set theory. However, the fuzzy modelling approach has a potential limitation that it requires a large amount of data to accurately train the model [Lawrynczuk, 2007a, Kuure-Kinsey et al., 2006b]. The errors from the unmodelled dynamics can be significantly reduced because of the ability of the neural network (NN) to model the complex or near impossible to model dynamic effects without using predefined analytical models. The learning of the complex dynamics mapping can be realised through the learning process from raw flight test data which simplifies the flight controller design significantly.

The ANN is a mathematical representation that mimics the biological neurons in the human brain. The typical tasks that are performed by biological neurons are shown in Figure 2.5. Each neuron in the network collects signals from other neurons through its dendrites and in turn sums and processes those messages within its Soma/Cell body. The processed signal is then transferred to the other neurons afterwards through the axon link and terminal buttons. Using a similar approach, the ANN model can be trained to produce solutions to modelling problems through a similar connection mechanism. The

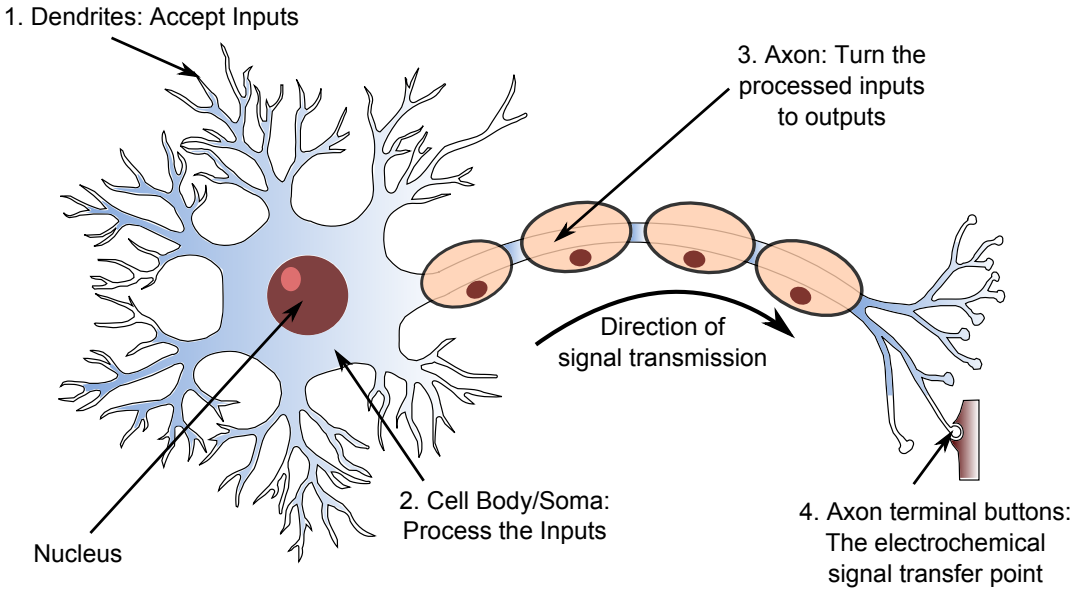


Figure 2.5 A simplified representation of a biological neuron.

NN architecture such as Multi-layered Perceptron (MLP) network has been theoretically proven to be an universal approximator, thus capable of approximating any non-linear function of interest to any desired degree of accuracy with sufficiently available neurons [Hornik et al., 1989]. Furthermore, the NN models are also able to establish the required input-output relationship without *a priori* assumptions about the properties of the data or the governing mathematical models as in first principle modelling [Abas et al., 2011].

The NN modelling approach has been used to perform a diverse range of tasks including prediction, function approximation, pattern classification and clustering. It has been proven in many research investigations and industrial applications as an efficient tool to handle complex input-output mapping. Paliwal and Kumar [2009] carried out a comprehensive literature review of over 100 comparative studies on the ANN and traditional statistical techniques used for prediction and classification tasks in various fields of applications. The review clearly highlights the potential and usefulness of the NN approach to approximate any non-linear mathematical function when it is difficult to handle the task statistically. It can be concluded from the review that NN models outperform the traditional statistical methods in most of the cases or at least performed as well as other statistical methods.

The literature also includes numerous applications of NN to address a wide range of

complex problems in aeronautical and flight control system applications. For example, the NN approach has been used to identify aerodynamic coefficients of an aircraft and rotorcraft system in Suresh et al. [2003], Dennis and Stengel [1992]. NASA Dryden Flight Research Centre and Boeing company have developed an adaptive flight control system that utilises the NN model to predict the stability and control parameters of the F-15 tactical jet fighter. This prediction data was then continuously used to optimise the control system and to assist the pilot in damage or failure situations of the aircraft [Urnes et al., 2001]. Furthermore, NN is also used to detect faults in the helicopter transmission system and has been proven reliable to classify faults even with different vibration signatures data [Parker et al., 1993]. Review papers commenting on the usage of the NN approach in the aeronautics and flight control system design have been contributed by Calise and Rysdyk [1998] and Faller and Schreck [1996]. There are two main advantages for transitioning neural network technology into UAS: (1) the potential to reduce the flight control system design and development costs, (2) possible reduction in cost associated with the development of large aerodynamic database [Calise and Rysdyk, 1998].

2.3.1 NN Model Structure Used for Dynamics Modelling

There exist several ANN model structures that have been used to model the dynamics representation of a UAV helicopter. The NN model structures differ from each other depending on how their processing units or neurons are interconnected. This section covers the main types of network structures used for modelling the helicopter flight dynamics.

Since the helicopter dynamics is non-linear, the NN system identification approach using the NNARX (Neural Network-Auto Regressive structure with eXtra inputs) model structure can be used to address such a problem. Here, the linear model structure such as the ARX model structure was introduced as the basic model structure while the NN network was used to introduce non-linearity to the model estimation. Similar to stability features of the ARX model structure, the prediction from the NNARX model was considered stable since there was only a pure algebraic relationship between

prediction and past input and output measurements [Norgaard, 2000].

In the NNARX model structure, the variables to be estimated and other influencing variables including their time lags are typically fed into a static feed-forward network such as the multi-layer perceptron (MLP) network [Samarasinghe, 2007]. Studies have shown that the NN based approach using the NNARX architecture is effective in modelling the dynamic response of the helicopter based UAS [Suresh et al., 2002, Samal et al., 2008, San Martin et al., 2006, Rimal et al., 2011, Chetouani, 2010]. Another interesting finding following the work of Rahideh et al. [2008] suggests that the NNARX networks trained with Levenberg-Marquardt (LM) algorithm is capable to exceed the prediction performance of first principle model.

Different neural network structures such as Radial Basis Function (RBF) and Recurrent Neural Networks (RNN) have been used in past research works with success in identifying the helicopter dynamics [Ahmad et al., 2002, Kumar et al., 2010, Pedro and Kantue, 2011]. An example of RBF network application for system identification of a twin rotor helicopter system was proposed in Ahmad et al. [2002]. The RBF network typically uses the orthogonal least square (OLS) algorithm to systematically find a set of weights and radial basis centres to ensure that the desired input-outputs mapping is performed [Chen et al., 1991]. However, the RBF network usually requires more hidden neurons compared with the conventional NNARX network with sigmoid or tangent hyperbolic activation functions in the hidden layer. The reason behind such an increase in the number of neurons used in the RBF network contributes to the fact that the outputs of the radial basis neurons only cover a relatively small input space compared to the outputs of typical activation functions in the NNARX networks [Shaheed, 2005]. Thus, in order to adequately represent broader input space, a much larger number of neurons need to be included in the network to achieve more reliable identification results.

Many parameters in a RBF network such as the network weights, biases, centre vectors and spread constant need to be optimised compared with a much simpler NNARX network that only require optimisation of weights and biases [Shaheed and Tokhi, 2002]. In terms of generalisation performance, NNARX network produces a much

better generalisation performance compared with RBF network but at the expense of much longer training time [San Martin et al., 2006]. Results show that NNARX network produces better global approximation for different flight manoeuvres compared with RBF network which produces a lower prediction error in certain flight manoeuvres.

The RNN architectures with dynamic memories have become a popular choice for system identification applications. They have the primary advantage in identifying dynamical system without prior knowledge about the model structure in contrast to NNARX or RBF network architectures, and incorporate dynamics information into the model using feedback from the output neurons (Jordan type networks) or the output of hidden neurons (Elman type networks) into the context units. These units are also known as the memory units which store past output values from the hidden or output neurons. The RNN architectures and their variant forms have been introduced into various system identification applications where the methods are found to be capable of representing a non-linear dynamical system with exceptional accuracy [Pham and Liu, 1993, Kalinli and Sagiroglu, 2006, Suresh et al., 2003, Samarasinghe, 2007].

Another type of recurrent NN model called Memory Neural Network (MNN) was introduced by Sastry et al. [1994] for identification and control of dynamic systems. This neural network modelling approach was later employed by Suresh et al. [2002] to model the longitudinal and lateral dynamics of a helicopter system. The operating concept of the MNN is quite similar to the Jordan and Elman networks as the MNN has several internal temporal memory units that are trainable to represent the dynamic systems without depending on the past output and input measurements. Each neuron in MNN is associated with a memory unit which stores the past values of the network neurons.

The RNN architectures suffer several drawbacks associated with the network's insufficient memory capacity which limits the prediction capability to lower system order. Several modifications have been suggested to increase the performance and memory capacity of the networks [Dong et al., 1994, Pham and Liu, 1993, Kalinli and Sagiroglu, 2006]. Moreover, Horne and Giles [1995] have shown that NNARX network performed better in some system identification problems than many conventional recurrent networks

and usually converged faster with better generalisation performance. The prediction performance of the NNARX network was also proved to be more suitably used for the identification of coupled helicopter dynamics compared with RNN architectures [Kumar et al., 2006]. The superior generalisation performance of NNARX networks was due to the introduction of embedded memory (tapped delay lines) to the network which reduced the NNARX network sensitivity to long term inputs-outputs dependencies [Lin et al., 1996].

Despite from various network architectures developed so far, the feed-forward Multi-layered Perceptron (MLP) networks such as NNARX and their hybrid variant [Mashor, 2000, 2004, Wilamowski, 2009] are still dominantly used for system identification and control purposes despite superior convergence properties of the RBF networks. This is due to their simple architecture and exceptional prediction performance in approximating complex non-linear mapping. Throughout the past few decades, intensive research has been conducted to improve the computation efficiency of the standard back-propagation algorithm and the training time of the MLP networks. The introduction of the modified back-propagation algorithms and several efficient second order methods such as the recursive prediction error (RPE) methods, Levenberg-Marquardt (LM) and neuron by neuron (NBN) algorithms significantly speed up the convergence time of the standard back-propagation algorithm [Riedmiller and Braun, 1993, Billings et al., 1991, Wilamowski, 2011a, Wilamowski et al., 2011].

Although the NNARX neural network is superior in terms of it prediction accuracy, the dynamic model identified from neural network can be inaccurate or wrong due to many problems arising from incorrect model structure selection, incorrect input vectors selection and over-fitting due to excessive number of neurons and insufficient training. Previous attempts to model the non-linear helicopter dynamics using the off-line NN modelling with tapped delay lines had successfully modelled the dynamics of the helicopter, resulting in low mean or standard deviation of residual values [Putro et al., 2009, Samal, 2009, Taha et al., 2010]. However, past efforts in identifying the helicopter dynamics did not include the effect of embedded memory or model order of the NNARX networks on generalisation performance for the modelling problem

considered.

NN based modelling approach that uses tapped delay lines to represent a dynamical system usually requires a priori knowledge about the model order of the system [Suresh et al., 2003]. Low model order assumption can generally lead to reduction of prediction performance of the neural network model. A much more in-depth analysis in selecting proper network structure needs to be performed, in order to improve prediction quality from the NN model. Obviously, the model validation plays an important part in the system identification steps. The validation methods introduced in this work can be used to ensure that the NN model fits well with observations and aid the neural network modeller to select the optimised or near optimised network structure for prediction with an acceptable accuracy. Furthermore, massive amount of weights in the standard NNARX model can results in increased NN training time and limit the NNARX model in real-time application. Several alternative architectures will be tested out in this study in order to improve the training time of the NNARX model.

2.3.2 NN Training Algorithms

There are numerous training algorithms available in the literature for NN training. The gradient based methods are common by used techniques to solve the minimisation problem in NN training. They are used to minimise the error cost between the measurement data and the predicted outputs of the NN model. Figure 2.6 shows the general principle of NN training or learning process. The minimisation process of the error cost function is carried out iteratively for a given data set to obtain a set of optimum NN weights. Subsequently, a set of properly trained NN weights gives the best possible fit to the measurement data. Several examples of gradient based methods commonly used for NN training are back-propagation technique (Steepest Descent Method), conjugate gradient method, Newton's method, Gauss-Newton (GN) method, Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, Neuron by Neuron (NBN) algorithm and Levenberg-Marquardt (LM) optimisation algorithm [Norgaard, 2000, Wilamowski et al., 2011, Wilamowski, 2011a, Haykin, 2009]. These NN training methods are considered as local minimisation techniques.

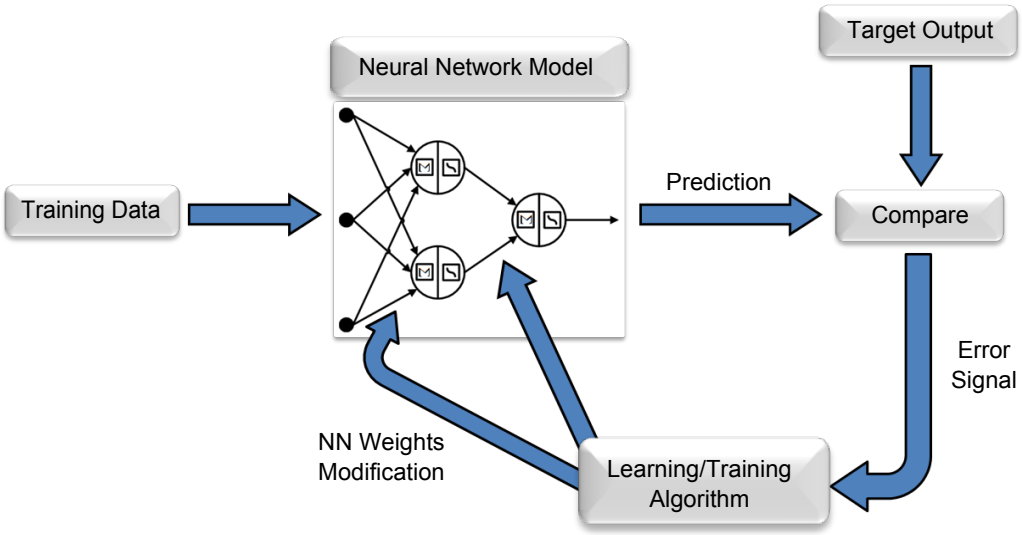


Figure 2.6 The representation of neural network training process.

The choice of training algorithm is an important factor to consider before the NN model is trained for prediction tasks. Empirical findings indicated that NN models trained with the back-propagation or steepest descent (SD) method were unable to generalise well with new observations and even failed to match the prediction quality of the first principle model. It is well known that steepest descent algorithm has several disadvantages compared with second order algorithms such as slow convergence rate; and the minimisation search can easily be trapped in local minima [Billings et al., 1991, Mashor, 2003, Wilamowski, 2011a].

Among all mentioned training methods, the LM optimisation algorithm is the most widely used training method for off-line NN training. The algorithm is capable of providing fast solution convergence, robust and simple to implement. The convergence of the LM algorithm is shown to be faster than other training methods [Samarasinghe, 2007, Garratt and Anavatti, 2012, Demuth and Beale, 2000]. However, the computation time of the LM algorithm increases as the size of the network grows. As the network becomes larger, the Jacobian and Hessian matrices that need to be calculated and stored in memory increase. Subsequently, this leads to increased computational time of the LM algorithm due to memory limitation [Wilamowski and Hao, 2010]. This prevents the LM algorithm from being implemented for large networks or modelling problems that involve an enormous size of training data.

Well known global optimisers such as the differential evolution (DE) algorithm can also be used to train the NN model [Ilonen et al., 2003]. The DE approach has the main advantage over the gradient based method such that the convergence to a global minimum is expected. Ilonen et al. [2003] comprehensively studied the effectiveness of the DE algorithm to find the global optimum in the context of the NN training. In this study, the DE algorithm was analysed as a candidate global optimisation method for the feed-forward MLP networks. However, the authors suggested that the DE algorithm did not provide any distinct advantages over the gradient based method in terms of learning rate or solution quality. This conclusion was also supported in the work of Subudhi and Jena [2011, 2008]. Results obtained envisage that the proposed NN training using the DE alone does not provide improved prediction capability and convergence speed of the training compared with the standard LM training. However, the authors suggested that a combination of DE and LM training algorithms provides better prediction accuracy which can be seen in the example cases provided. It is noticed from the result that the convergence time for the proposed training method (DE+LM+NN) is still slower than the NN model trained with the standard LM algorithm. Subsequently, the conclusion can be drawn that the DE training is unattractive for the real-time system identification framework.

Most NN based modelling techniques attempt to model the time varying dynamics of a UAS helicopter system using off-line modelling approach. The model which is generated and trained once from previously collected data is not able to represent the entire operating points of the flight envelope very well [Samal, 2009, Ljung and Soderstrom, 1983]. Several attempts such as Samal [2009], Samal et al. [2008, 2009] were made to update the NN prediction model during flight using mini-batch LM training (LM training with small number of data samples). However due to a limited amount of processing power available in the real-time processor, such methods can only be employed to relatively small networks and they are limited to model uncoupled helicopter dynamics.

In order to accommodate the time-varying properties of helicopter dynamics which change frequently during flight, a recursive based learning algorithm is required to

properly track the dynamics of the system under consideration. Furthermore, the usage of recursive algorithms such as recursive Gauss-Newton (rGN) or recursive Levenberg-Marquardt (rLM) reduces the computation complexity of the off-line (batch) training method without having to invert the full Hessian matrix in every iteration [Ngia and Sjoberg, 2000]. The implementation of the recursive training using rGN algorithm is proposed in this work and compared with mini-batch LM training to determine the effectiveness of recursive training for real-time NN prediction.

2.4 AUTOMATIC FLIGHT CONTROL SYSTEM

Automatic flight control system (AFCS) for helicopter based UAS was traditionally designed using linearisation of rigid body equation of motion at various points throughout the flight envelope [Mettler et al., 2000]. Since we have a collection of linear models that represents a particular flight regime, the gain scheduling approach is the commonly used method to design flight controllers for UAS. The overview of the gain scheduling control approach is shown in Figure 2.7. In this approach, several linear models are obtained from the non-linear model in certain flight conditions. Subsequently, linear controllers are then designed for each different flight conditions such as hovering or forward flight at different velocities. Since we have multiple linear controllers that provide satisfactory control for different operating points, the gain scheduling approach is used to determine the current flight operating region and to activate the appropriate linear controller. Several measured variables from the on-board instrumentation such as airspeed, dynamic pressure or altitude can be used to trigger specific linear controllers relative to the current flight operating region.

The complex, inherently unstable and non-linear nature helicopter based UAS present a serious challenge for design and implementation of a control strategy. Typically, the control system architectures for helicopters were primarily based on the stability augmentation systems (SAS), which are concerned with attitude or altitude control and stabilisation. Besides SAS, the AFCS developed for helicopter based UAS can also include other types of control loops including velocity/position control, heading control and 3D trajectory tracking. Throughout the past few decades, a wide range

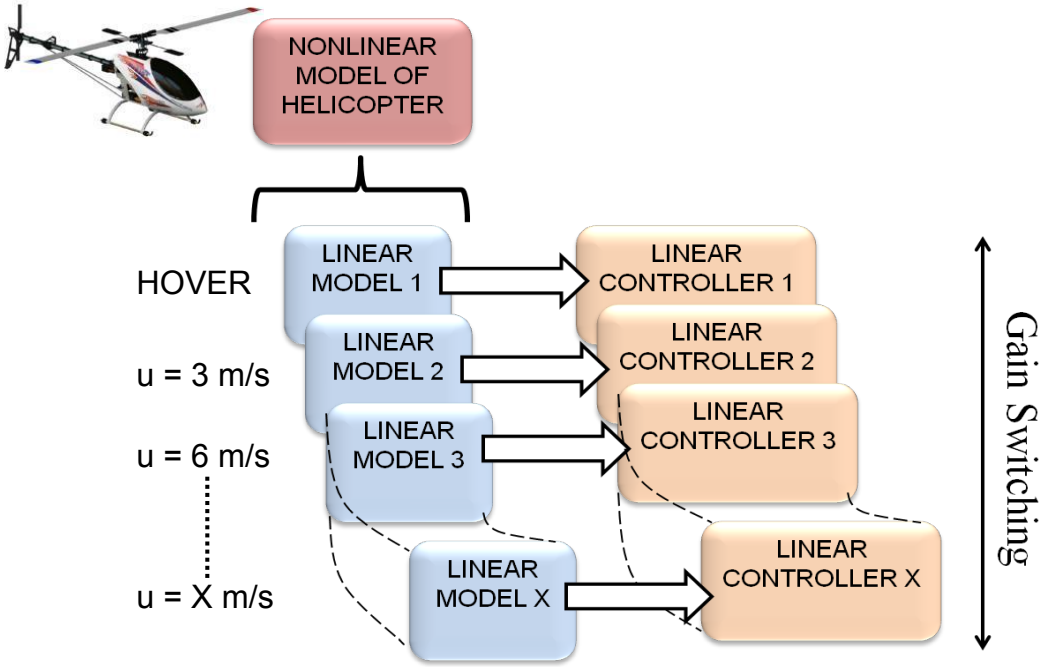


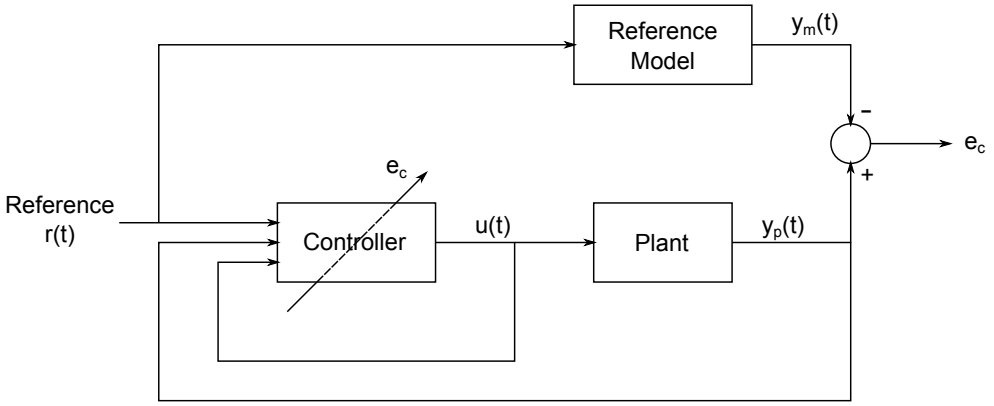
Figure 2.7 An overview of gain scheduling control approach.

of control methodologies were proposed to address helicopter control problems, which vary from the classical control method to learning based control. Detailed review on the existing control methodologies applied to the unmanned helicopter system is given in Kendoul [2012], Nonami [2010], Valavanis [2007]. These flight controllers are generally classified in three broad categories such as linear flight controllers, model based non-linear controllers and learning based flight controllers.

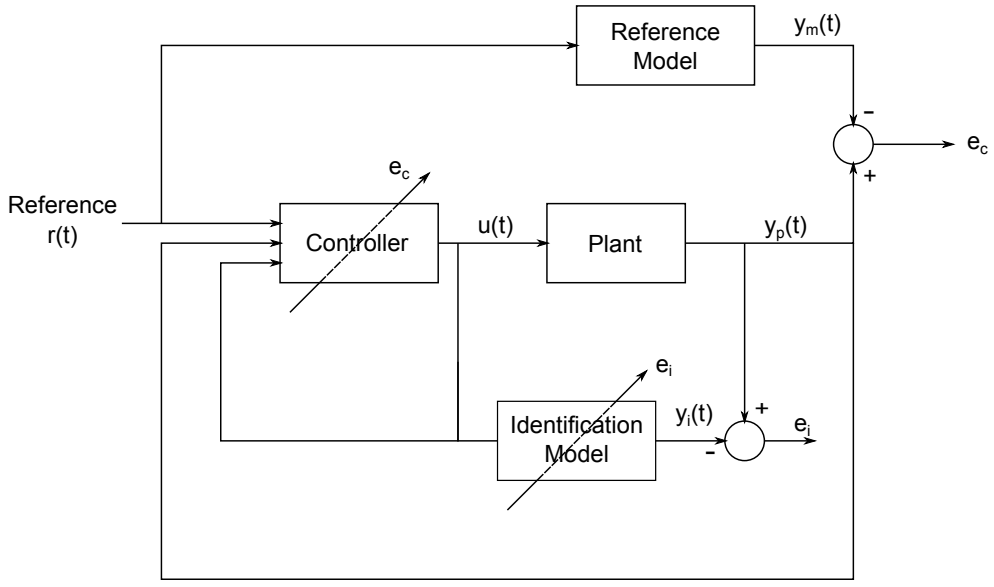
The linear controllers are currently the most favoured methods used by military and industrial research communities due to the simple implementation and intuitive control system structure. There exist many available tools that can be used to refine the controller's gains and to analyse their performance and robustness. Moreover, the linear controllers have been successfully implemented in UAS applications and have been shown to be effective in numerous flight tests [Mettler, 2003, Valavanis, 2007, Saripalli et al., 2003, Amidi et al., 1999, Kendoul, 2012]. Recent research on helicopter UAS has focussed on the use of linear control theories such as Linear Quadratic Regulator [Shin et al., 2005, Nonami, 2010, Bergerman et al., 2007], H_∞ control [Civita, 2003, Walker, 2003] and Gain Scheduling control [Antunes et al., 2010].

The desire for enhanced agile manoeuvres and accurate tracking control requires the unmanned helicopter system to be operated beyond the range of nominal operating conditions. In order to extend the capabilities and operating range of the linear controllers, the non-linear dynamics of the helicopter UAS can be linearised into sets of linear models each representing certain operating condition. The tuning of the controller gains for each linear condition is done via interpolation schemes using the gain scheduling approach. However, such a technique has been reported to suffer performance degradation when performing large amplitude manoeuvres [Kendoul, 2012, Mettler, 2003, Valavanis, 2007]. In addition to limitation of linear approaches, the control problem becomes much more challenging to solve as the helicopter operates under varying atmospheric disturbance.

Although the gain scheduling approach is widely used in unmanned helicopter control, the tuning of controller's gains can also be difficult due to the presence of faults in the system and non-linearity in the operating conditions [Vijaya Kumar et al., 2011]. This specific problem in the gain scheduling control can be overcome by considering recursive adaptive control methods. The adaptive controller is categorised under model based non-linear control where the control parameters are updated at every time step. This control strategy is implemented to rapidly respond to the varying flight conditions or component failure scenario. Under the adaptive control approach, the controllers are divided into two types of control configurations [Samal, 2009, Narendra and Parthasarathy, 1990]: (1) direct adaptive controllers, and (2) indirect adaptive controllers. Figure 2.8 shows the basic configuration structure of the direct and indirect adaptive control system for a dynamic system. In direct adaptive controllers, the control parameters are updated directly to minimise the tracking error. The calculation of the controller parameters or gains does not rely on the dynamics system to update the controller parameters. Whereas in the indirect adaptive control configuration, a dynamic model is used to predict the response of the dynamic plant. The prediction from the identification model is assumed to be identical to the actual response and this information is used to minimise the tracking error of the system.



(a)



(b)

Figure 2.8 The configuration of adaptive control system: (a) direct adaptive controller; and (b) indirect adaptive controller.

2.4.1 Neural Network Based Control Design for Unmanned Helicopter System

In recent development of real-time adaptive control system, the neural network approach has gained popularity in the development of AFCS due to its ability to learn complex mapping from flight data sets. The NN calculation is parallel in nature, which leads to faster calculation speed in an intensive computation problems [Balakrishnan and Weil, 1996]. Furthermore, the NN has the ability to adapt well to varying dynamic properties which make it suitable for adaptive control application. Typical neural network based

control schemes can be categorised into six main classes as follows [Norgaard, 2000]:

1. Direct inverse Controller
2. Internal model Controller (IMC)
3. Feed-forward controller with inverse model
4. Feedback linearisation controller
5. Optimal controller
6. Non-linear Model Predictive Controller (NMPC)

Samal [2009], Norgaard [2000], Agarwal [1997] suggested that the first five types of NN controller schemes fall under direct adaptive control class where the NN is used to update the controller parameters. Whereas, the NMPC is an indirect type of adaptive controller where the NN model is used to aid the existing conventional MPC controller to drive the system response towards the desired reference trajectory.

One of the earliest NN methods used to control the non-linear flight dynamics is the direct inverse controller approach. In this approach, the NN was trained to act as an inverse of the system. This results in a NN representation that directly maps the sensor inputs (past output and input measurements) to actuator controls. This representation was later used as a controller for the system. Mathematically, the inverse model of the system is described as follows:

$$u(t) = f^{-1}\left[y(t+1) \quad y(t) \quad \cdots y(t-n_y+1) \quad u(t) \quad \cdots \quad u(t-n_u)\right] \tag{2.3}$$

After the NN controller has been trained, the inverse model can be used to control the system by substituting the output at time $t+1$ with the reference set-point $r(t+1)$ [Norgaard, 2000]. The inverse model can be trained either using the off-line training or on-line training methods used in system identification problem. The general principle of the direct inverse controller is illustrated in Figure 2.9.

In Buskey et al. [2001], a direct inverse type NN controller was designed and developed for automatic hovering of the JR Ergo RC helicopter. The weights of the

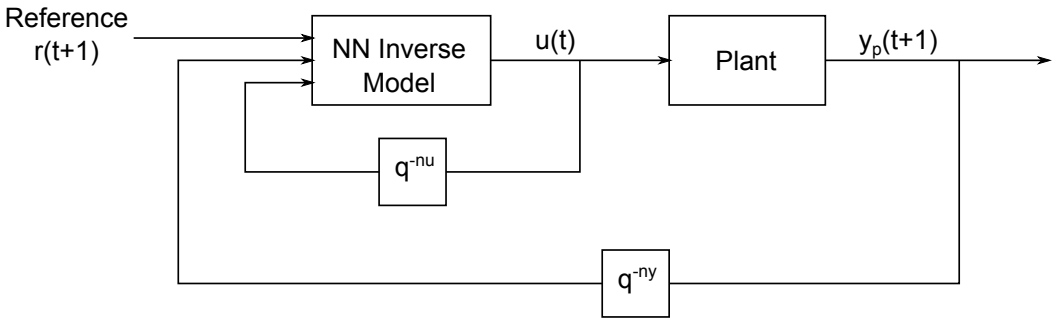


Figure 2.9 The NN based direct inverse control.

inverse NN model were determined using the off-line back-propagation training method. Partial hovering for several seconds was achieved for the RC model helicopter. Although the training of the NN controller was intuitively simple and easy to implement, the off-line training method used was found to be unsuitable for tracking time varying dynamics of the helicopter. This was evident from the inability of the NN controller to properly regulate the helicopter roll motion under different flight conditions. To ensure the direct inverse NN controller to perform satisfactorily, several important issues such as proper excitation, optimal model structure selection and over-training need to be properly considered in relation to the off-line training method [Norgaard, 2000, Shamsudin and Chen, 2012a]. Furthermore, Kendoul [2012] pointed out in his survey study that the stability and robustness of the NN based methods are difficult to analyse.

Recently, Kumar et al. [2009] proposed a direct inverse type NN controller for an unstable helicopter system. The controller was capable of tracking the pitch rate reference signal generated using a reference model. The NN controller used in this work utilised the same NNARX network architecture which was commonly used in the system identification problem. The controller parameters (weights of the inverse model) are approximated initially using the back-propagation through time (BPTT) algorithm. To ensure the stability of the closed loop system, the NN controller's network parameters are adapted recursively (on-line) using the Lyapunov approach. Results indicated that the proposed NN controller for pitch rate channel demonstrate the ability to adapt to the parameter uncertainties such as control surface faults and aerodynamic uncertainties. The proposed NN controller design was later extended for roll and yaw controls in

Vijaya Kumar et al. [2011]. The theoretical results presented are validated using a non-linear 6 DOF helicopter simulation undergoing several agile flight manoeuvres. The NN controllers developed are found to perform well in the presence of gust, sensor noise and modelling uncertainty. Moreover, the NN controllers are shown to meet the frequency domain and phase delay requirements of the U.S. Army Aeronautical Design Standard for rotorcraft handling qualities (ADS-33E-PRF). Although the stability analysis has been developed for such a controller, no extensive experimental evaluation has been performed yet to test the performance of the NN controller.

In Suresh and Sundararajan [2012], a feed-forward adaptive NN controller scheme was proposed for a helicopter performing highly non-linear manoeuvres. The feed-forward NN control strategy used a NN controller to aid a basic LQR controller as shown in Figure 2.10. The NN controller is based on the on-line learning RBF network which used the Lyapunov based rules update to achieve global stability and better tracking performance. The pre-training and prior selection of the number of hidden neurons that accurately represent the inverse model are not required for the on-line learning RBF network. Instead, the appropriate number of neurons is determined recursively through the use of neurons growing and pruning algorithms. In the growing algorithm, the neurons of the inverse model are allowed to grow based on the corresponding error signal thresholds. In order to reduce the complexity of the NN model, the pruning strategy is incorporated in the algorithm to delete the non-contributing neurons. The simulation studies carried out in this work used a non-linear 6 DOF helicopter model similar to BO-105 helicopter. The performance simulation results clearly show the superior handling qualities of the proposed NN controller at various flight speeds and operating conditions. The results also indicate that the proposed on-line learning NN controller was able to adapt to the dynamic changes and provide the necessary tracking performance in executing highly non-linear manoeuvres such as obstacle clearance and elliptic manoeuvres.

The NN based adaptive linearisation approach is in principle similar to standard linearisation controller used in Isidori [1995] and Slotine and Li [1991]. Figure 2.11 shows the general implementation of NN based feedback linearisation approach. As shown in

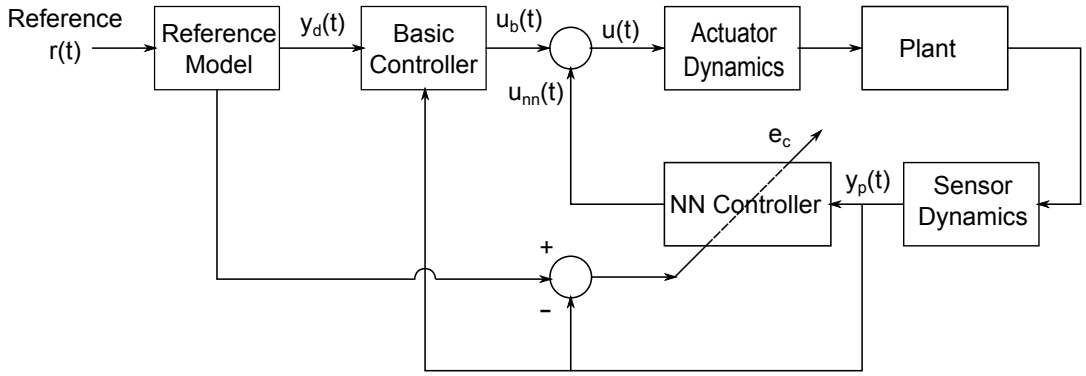


Figure 2.10 The general schematic diagram of an on-line training feed-forward NN based control where a neural network is added to the existing control system.

the figure, the control input signal of the feedback linearisation approach consists of two signal components where the first component cancels out the non-linearities in the system while the second component is a linear state feedback controller that stabilises the system [Hagan and Demuth, 1999]. The first component of the control signal can be approximated using the NN approach.

Kutay et al. [2005] implemented an adaptive output feedback control that employs feedback linearisation for a 3 DOF helicopter model control. This design approach permits adaptation to both parametric uncertainty and un-modelled dynamics. It also allows adaptation under known actuator characteristics including actuator dynamics and saturation. The control design approach employed in this work was implemented together with an on-line trained NN model that compensates for modelling error and a linear compensator that filters the tracking error. The control design was tested to control the pitch motion of the 3 DOF helicopter model using attitude feedback information obtained through a low resolution optical sensor.

In Nakanishi and Inoue [2002] and Nakanishi et al. [2002], a NN based feedback linearisation controller was used to control the Yamaha RMAX helicopter. The parameters of the NN controller are trained using recursive type NN training method such as Powel's Conjugate Direction algorithm. The performance of the proposed NN controller was tested with the use of Yamaha RMAX flight simulator and validated flight experiments. Four SISO type flight controllers (roll, pitch, yaw and altitude controllers) were independently designed and implemented in the flight experiments.

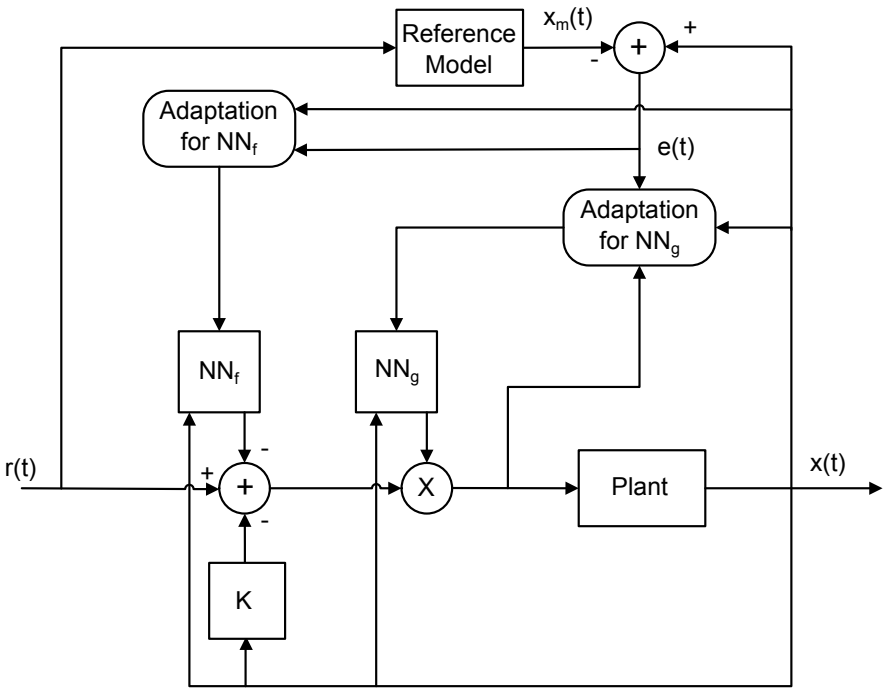


Figure 2.11 The general schematic diagram of a NN based feedback linearisation approach. Figure adapted from [Hagan and Demuth, 1999].

The flight experiment results suggest the effectiveness of the approach to track the reference commands properly, and be able to compensate for the undesirable effects of the un-modelled and time varying dynamics of the helicopter UAV.

Generally, the NN based optimal control approach involves designing a controller according to a criterion where the trajectory tracking is minimised while incorporating penalty on the magnitude of the control input. In this approach, the NN model is used to provide the control inputs that minimises this criterion. In Nodland et al. [2012], an optimal controller design was proposed for tracking control of an unmanned helicopter system using an adaptive critic NN framework. In this approach, an on-line approximator based dynamic controller was used to learn the continuous time Hamilton-Jacobian-Bellman (HJB) criterion equation and formulate an optimal control input that minimised the HJB criterion forward in time. The optimal controller in this approach consists of a single NN model which is tuned on-line using a novel weight updating law while maintaining the closed loop system stability. The stability analysis and performance validation of the proposed controller are done in Simulink® simulation for various flight manoeuvres such as automatic take-off, landing and hovering at certain

altitude. Simulation results confirmed that the proposed controller scheme is able to provide a reasonable tracking performance.

2.4.2 Neural Network Based Model Predictive Control

Model Predictive Control (MPC) or also referred as the receding horizon control (RHC) is an advanced control technique that relies on prediction from a process model, optimisation and receding horizon implementation. In more detailed definition of MPC, Mayne et al. [2000] suggest that the MPC is a form of feedback controller in which N sequence of control actions are obtained by minimising a finite horizon, open-loop optimisation problem over a prediction horizon of P steps $(k, k+1, k+2, \cdots, k+P)$. The solution of the optimisation process is done at each sampling instant subject to hard constraints on controls and states, using the current state of the plant as the initial state. The results of the optimisation process produce N optimal control sequences where only the first control action from this sequence is implemented to the dynamic plant and the process is repeated. The general concept of MPC algorithm is best described in Figure 2.12 which illustrates the behaviour of control input and system output changes in the MPC framework. In this figure, y_k indicates the output or state measurement, $\hat{y}_{k+P}$ is the predicted output from the internal MPC model over the prediction horizon, u_{k+N-1} is the predicted control input and N denotes the finite future horizon steps. The MPC technique has been widely used in the petrochemical industry and currently, the MPC usage has been extended for automotive and aeronautical application. The MPC ability to handle constraints on the manipulated inputs has been identified as one of primary reasons for the implementation successes of MPC in industrial applications. The MPC algorithms are also more flexible in terms of objective function formulation, time delays and multi-variable process control handling, which subsequently attracts more attention for their implementation in industries. Several application examples of MPC in automotive or aeronautical applications can be found in Castillo et al. [2007], Dalamagkidis et al. [2010], Joelianto et al. [2011] and Dahunsi and Pedro [2010]. Several survey papers have been published to overview the MPC histories, theoretical development, practical application and critical issues regarding MPC implementation

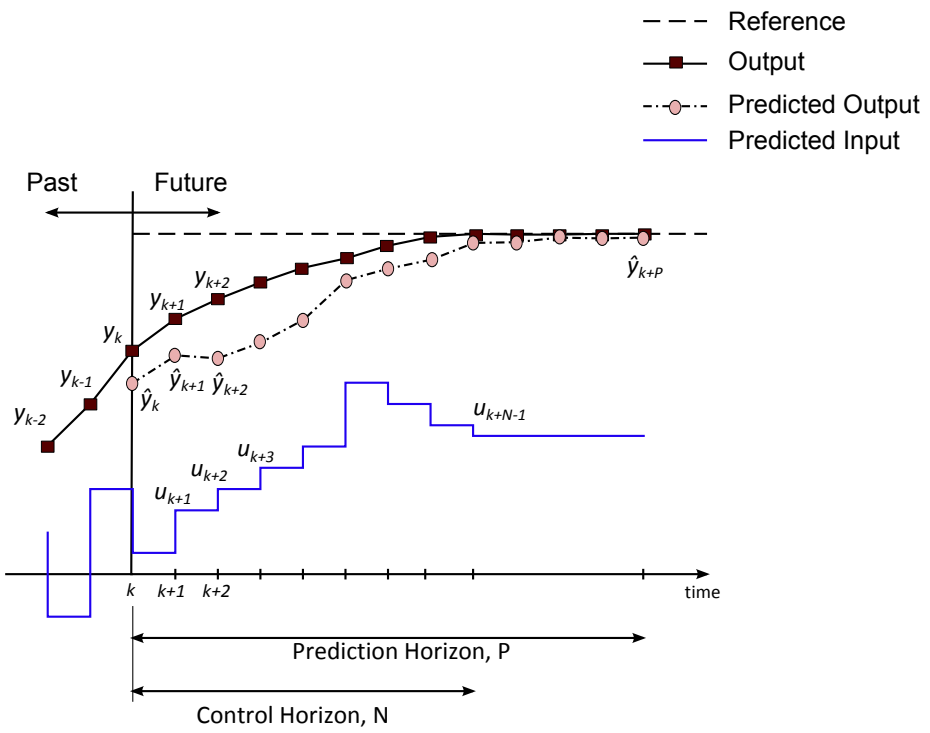


Figure 2.12 The implementation concept of model predictive control (MPC).

such as modelling, identification, robustness, state estimation, stability and optimality [Camacho and Bordons, 2007, Mayne et al., 2000, Rawlings, 2000, Bequette, 2007].

The primary objective of the MPC is to formulate a trajectory of future manipulated inputs to optimise the future behaviour of the dynamic system. The optimisation process is done in a fixed size moving horizon window which differs from other traditional control techniques that use pre-calculated control law and do not explicitly consider the future implication of the current control action [Mayne et al., 2000]. In order to bring the future or predicted system output as close as possible to the reference signal, the optimisation of the manipulated control input needs to be implemented recursively using the current state measurement of the plant. The future behaviour of the plant is predicted using a process model obtained either through mathematical modelling or system identification approach.

The linear MPC design generally utilises linear models such as the finite impulse response (FIR)/step response models, transfer function model and state space model [Rossiter, 2003, Wang, 2009b]. Wang and Young [2006] reports that the application of FIR models is limited to stable plant and requires a large model order, whereas the

transfer function models is applicable for both stable and unstable plants. Typically, the state space model is preferred over other models in the MPC controller design because of its simplicity and effectiveness in handling multi-variable processes. The linear models are widely used with MPC due to relatively simple procedure to identify the plant dynamic through system identification methods. They generally give good results whenever the plant is operating within or near the operating flight condition. Moreover, the application of linear models with MPC objective function results in a quadratic and convex optimisation problem which is much easier to solve compared with the usage of non-linear dynamic models with MPC. Unique solution of MPC optimisation with linear models enables direct implementation of the control law to meet real-time requirements.

In many control applications, the dynamics of the system under consideration is non-linear with frequent changes from one condition to another. Therefore, a linear model of the system is inadequate to represent the broad range of operating conditions and this motivates the application of non-linear dynamic model with MPC to achieve better control performance. The application of non-linear model for prediction process in MPC leads to non-convex and non-quadratic optimisation problems [Lawrynczuk, 2007b]. Subsequently, the nature of the optimisation problem can lead to solution with multiple local minima in the objective criterion as shown in Figure 2.13. Lawrynczuk [2007b] further argues that for a non-linear optimisation problem, there are currently no reliable and fast optimisation algorithms that are able to determine the global optimal solution at each time step within a prescribed time limit. The gradient based optimisation algorithms such as those suggested in Norgaard [2000], Wilamowski [2011a], Haykin [2009] may trap in local minimum and give no guarantee that a global optimal solution is found.

Another way to represent the broad range of flight operating conditions is by using different multiple linear models for different flight modes as suggested in Joelianto et al. [2011] and Gopinathan et al. [1998]. The multiple linear models are obtained by linearising the non-linear dynamic model from first principle modelling for each trim condition of the helicopter flight mode. The design parameters of the MPC controllers

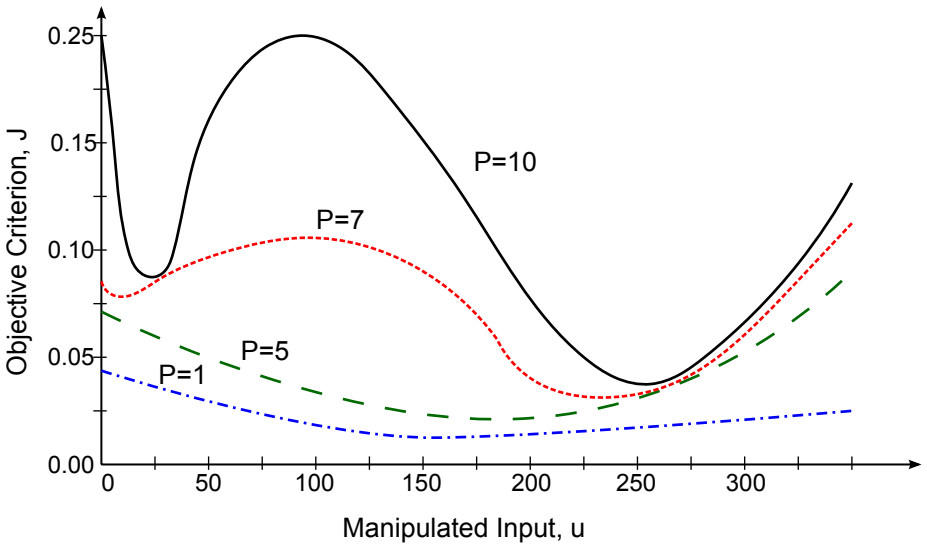


Figure 2.13 The illustration of non-convex optimisation with multiple local minima. The optimisation solution became trapped in local minima after prediction horizon increased greater than 5. Figure adapted from [Bequette, 2007].

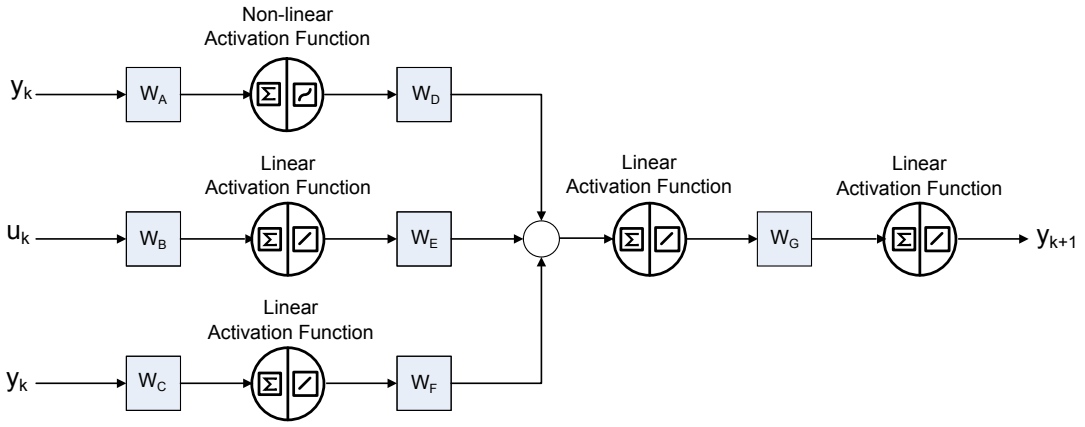
are then fine tuned appropriately for each linear model. The proposed strategy is similar to the strategy employed in a gain scheduling controller. Joelianto et al. [2011] reports that the MPC controllers are capable of handling the transition between linear models with exceptional control performance. However, the suggested controllers can result in a control signal that jumps to different values at each model transition. This effect needs to be reduced in practice as the sudden increase in the control action can result in dangerous sudden movement of the helicopter.

The neural network based ARX (NNARX) model is a popular non-linear model structure that can be used with the MPC algorithm [Norgaard, 2000, Soloway and Haley, 1996, Norgaard et al., 1996, Sadhana Chidrawar and Waghmare, 2009]. In Samal [2009], a NN based MPC was developed for an unmanned helicopter UAS that takes into account the constraints and non-linearity of the helicopter dynamics. The NN model used in the MPC formulation and system identification is based on NNARX architecture. The optimisation of the NN based MPC approach is solved using the non-linear optimisation technique. The performance of the proposed controllers is validated in simulation and real flight tests. Numerical simulation results show the robustness of the proposed controller under the effects of actuator and sensor delays, measurement noises, wind gusts and possible parameter uncertainties. Furthermore,

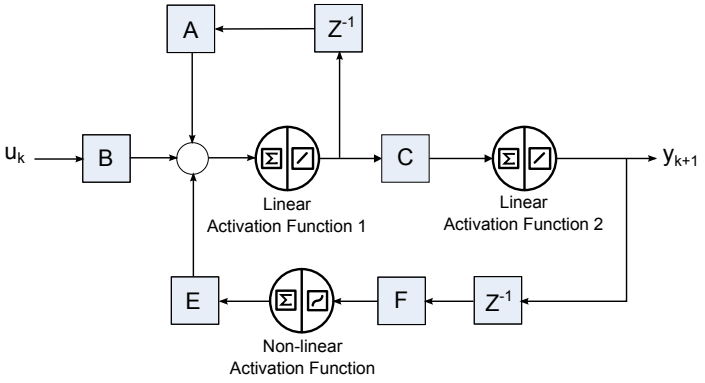
results from the flight tests provide further justification of the proposed controller. However due to demanding computation of the non-linear optimisation of the proposed method, only SISO based controller was implemented in flight. Further simplification is made to the optimisation problem where only soft constraints are considered in the work. The hard constraints on the amplitude and the rate of the control inputs are considered only as saturation and rate limiter blocks. The limitations due to the high demanding computation of the non-linear optimisation should be addressed, in order to enable us to fully utilise the capability of the MPC technique.

The implementation of the NNARX model with MPC algorithm involves high computation effort since the non-linear optimisation problem needs to be solved at each sampling time. This causes the control implementation to be only feasible with slow dynamic processes [Allgower, 2000, Norgaard, 2000]. Several alternatives to solve the non-linear optimisation problem is reported in Lawrynczuk [2007a]. Lawrynczuk [2007a] proposes that the straightforward and simple solution to the non-linear optimisation problem can be achieved by using the approximation of the non-linear model used in the MPC to resemble linear form that nearly matches the system under consideration.

The idea to use approximation to the non-linear model such as NN to mimic the simpler linear model form has been proposed in Kuure-Kinsey et al. [2006a], Kuure-Kinsey and Bequette [2008] and Norgaard [2000]. In these published work, a novel MPC approach is developed to control the 'van de Russe' reactor using the feed-forward NNARX or RNN model. The approximation to the non-linear feed-forward NN model is obtained by decoupling the standard feed-forward NNARX network inputs into separated groups of inputs as shown in Figure 2.14(a). The modification to the standard NNARX network produces a NN model with a 3-layer architecture. Similarly, the same arrangement can also be made with the RNN architecture as shown in Figure 2.14(b) which results in a 2-layer network. As can be seen in the figure, the RNN network is comprises two recurrent layer connections from the output, y_{k+1} to the input of layer 1 and from the output of layer 1 back to the input of layer 1. The calculation of the predicted outputs from the feed-forward NNARX and RNN model results in non-linear state space model representations. The linear state space models



(a)



(b)

Figure 2.14 Different configuration of NN architectures that represent time varying linear model [Kuure-Kinsey et al., 2006a, Kuure-Kinsey and Bequette, 2008]: (a) Feed-forward NN architecture (b) Recurrent NN architecture

are obtained at each time step by linearising the non-linear path of the non-linear state space model with a first order Taylor series expansion. The standard MPC formulation is then devised for the control problem using the linearised models obtained from the NN. The simulation results presented in both papers point out that the performance of the proposed controller is comparable to the non-linear MPC with good robustness performance against disturbances. Furthermore, the proposed NN based MPC requires less computation time to execute compared with the non-linear NN based MPC. Better controller performance is also achieved with RNN architecture with significant improvement compared with the feed-forward NNARX model [Kuure-Kinsey and Bequette, 2008].

The NN based predictive control proposed in Norgaard [2000] is quite similar to the

approach taken by the above-mentioned approaches. Norgaard [2000] proposes that the high computation effort involving MPC implementation with non-linear NNARX model can be reduced with the so-called 'instantaneous linearisation' technique. This technique is employed to extract the linear transfer function models at every time step from the non-linear NNARX model. No modifications are made to the NNARX model structure, in contrast to the approach taken by Kuure-Kinsey et al. [2006a]. Similar to above-mentioned approaches, the extraction process of the linear models from the NNARX model enables the execution of linear MPC strategy, which is simpler in terms of finding the minimum of the controller's optimisation criterion. The extracted linear models from the NNARX can also be applied for various linear controller techniques such as pole placement method and minimum variance design [Norgaard, 2000, Ab Wahab et al., 2009]. Furthermore, the usage of the instantaneous linearisation principle provides more valuable information about the dynamics of the system compared with the black-box nature of the NNARX model.

The transfer function of MPC introduced in Norgaard [2000] is often regarded to be less effective in handling multi-variable dynamic processes [Wang, 2009a]. Transformation of the transfer function form into a non-minimal state space (NMSS) formulation should improve the proposed NN-MPC formulation in terms of simpler design framework and analysis. Furthermore, the NMSS form still retains the basic features of the transfer function model which only requires state variables chosen from the set of measured outputs, inputs and their time lag values. This modification to original transfer function form can avoid the use of an observer to access the state information.

Since the dynamic of the helicopter based UAS is part of a highly complex system with strong non-linearity, designing an automatic flight control for such a dynamic system is difficult and presents significant challenges. Moreover, the dynamic system of the helicopter is also known to be coupled and time varying, which requires necessary compensation through the use of adaptive type controllers. Numerous types of direct adaptive NN controllers have been reported in the literature with the majority of the control techniques being only validated in numerical simulations. More extensive experiments need to be performed to evaluate the effectiveness of these approaches in a

wide range of environments. Although the implementation of NN based direct adaptive controller is simple, the design and tuning of the controllers are difficult, requiring a retraining of the NN controller each time the controller parameter is altered [Norgaard, 2000].

The indirect adaptive type controller such as NN based MPC is much more flexible in implementation where only the model is identified by the NN while the controller action (MPC) is calculated at each time step. The MPC controller with its anticipated benefits have been employed successfully to control helicopter based UAS and is expected to work well with the desired control performance. The NN based system identification techniques reviewed in the previous section are suitable to be used to identify the dynamic plant. Considering the convergence and computational speed issues of the NN based MPC, an approximation based NN-MPC can be used to speed up the computation speed. Based on the findings from the literature, the NN based MPC seems to fit our common goal to develop a general adaptive flight controller that can be developed with less time and effort.

2.5 SUMMARY

In this chapter, a brief literature review about the definition, application and classification of the rotorcraft UAS is provided as a basic introduction to the research topic. An overview of various autonomy components that enable the rotorcraft based UAS to perform autonomous missions is presented. This research project will focus mainly on the development of automatic flight control components since the automatic control component is the first key component that needs to be addressed in order to enable a UAS to be autonomous.

A brief literature review on the helicopter dynamics modelling and NN based system identification is presented. Various factors that influence the NN prediction and training performance such as the model structure selection and the choices of training algorithms are discussed and highlighted. Available literature on the different AFCS design techniques is explored with special emphasis given to flight control system

design based on the NN approach. In the following chapter, the helicopter platform and avionic systems used for development and validating the system identification methods and control algorithms are described in detail, along with the introduction to the development of the safety test rig for the unmanned helicopter system.

Chapter 3

AERIAL PLATFORM AND CUSTOM-BUILT FLIGHT TEST SYSTEM

3.1 INTRODUCTION

In this chapter, the helicopter platform and avionic systems used for development and validation of the system identification methods and control algorithms are described in detail. Two avionic systems had been developed specifically for system identification and flight control testing. For flight control testing, a safety test rig was developed to restrain the helicopter movement which could prevent fatal crashes due to possible hardware failures or programming errors.

The chapter is organised as follows: In Section 3.2, the aerial vehicle platform used for flight testing of the system identification and control methods is presented. Next, a description of test stand development is provided in Section 3.3 for safe testing of the proposed flight controller system. The test stand design consists of two main sections to simulate the translational and rotational motion of the helicopter. These features provide a full 6 degree of freedom (DOF) flight function to the test stand design. Then, the overall architecture of flight instrumentations, ground control station (GCS) and avionics system for automatic flight controller development are given in Section 3.4. The system overview for system identification experiment is given in Section 3.5. Finally, the chapter is summarised in Section 3.6.

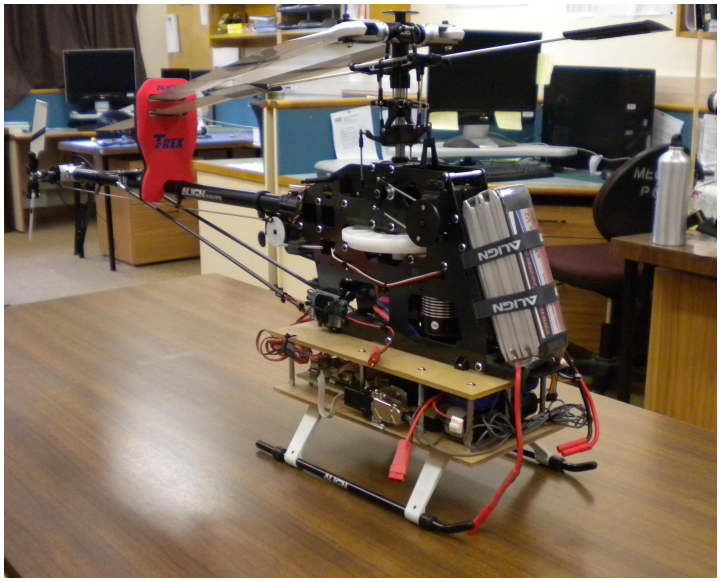


Figure 3.1 The TREX600 helicopter used in this research project with instrumentation equipment fitted between the fuselage and the landing gear.

3.2 AIR VEHICLE DESCRIPTIONS

The unmanned aerial vehicle (UAV) platform which was used in this research is a conventional electric model helicopter known as TREX600, manufactured by ALIGN Corporation. The helicopter model was selected due to its sufficient payload capacity, great manoeuvrability and low cost replacement parts. It was equipped with a standard Bell-Hiller stabiliser bar on the main rotor, which improves handling characteristics for human pilots by increasing the damping on the pitch and roll responses. Furthermore, TREX600 was also equipped with a high efficiency high torque brushless DC motor that allows the helicopter to carry about 2 kg payloads with an operation time of about 15 min. The basic UAV platform shown in Figure 3.1 had been modified to make room for installing necessary electronic equipment which gathers flight data for dynamic modelling and control system design. Some key physical parameters of TREX600 RC helicopter are given in Table 3.1.

This helicopter consists of a fuselage, a main rotor, a tail boom/tail rotor assembly and landing skid. The main rotor and tail rotor are driven by an electric brushless DC motor which is mounted under the transmission casing for compact design. This prohibits the mounting of any avionic systems in the area underneath transmission

Table 3.1 The specification of TREX600 ESP helicopter.

Specifications	TREX600 ESP
Length	1.16 m
Height	0.4 m
Main Blade Length	0.6 m
Main Rotor Diameter	1.35 m
Tail Rotor Diameter	0.24 m
Weight	3.3 kg
Endurance	15 min

casing and leaves this vehicle less attractive for tight component installation.

The main structure of the helicopter consists of two vertically mounted parallel main frames made of carbon fibre, battery and gyro mounts (plastic), metal motor mount and a metal bottom bracket. This simple structure design produces minimum weight with maximum strength for the helicopter platform. The electric motor and associated reduction gearing are mounted in between these two carbon fibre main frames, with various accessories and control components attached where appropriate as shown in Figure 3.2. The factory also provided a lightweight plastic landing skid which was connected to the metal bottom bracket via four mounting points.

The RC helicopter usually has a very high rotor speed of around 1500 rpm and fast dynamic response due to its small inertia value. Shim [2000] had reported that in order for scaled model helicopters to achieve equilibrium of lift on the rotor disc in less than one rotor revolution, most of the small size helicopters would require stabilisation response time to be achieved in less than 40 ms. Without any extra stability augmentation devices, this is an extremely short time for radio controlled (RC) pilots to control the helicopter. For this reason, almost all small-size RC helicopters have a mechanism to artificially introduce damping. In most RC model helicopters, a large control gyro with an air-foil, referred to as a stabiliser bar or fly-bar is used to improve the stability characteristic around the pitch and roll axes and to minimise the actuator force required to control the main rotor cyclic pitch [Kim and Tilbury, 2004]. In addition, an electronic gyro is also used with the tail rotor input command to further stabilise the yaw axis.

According to Mettler et al. [2002a], the stability augmentation system can be

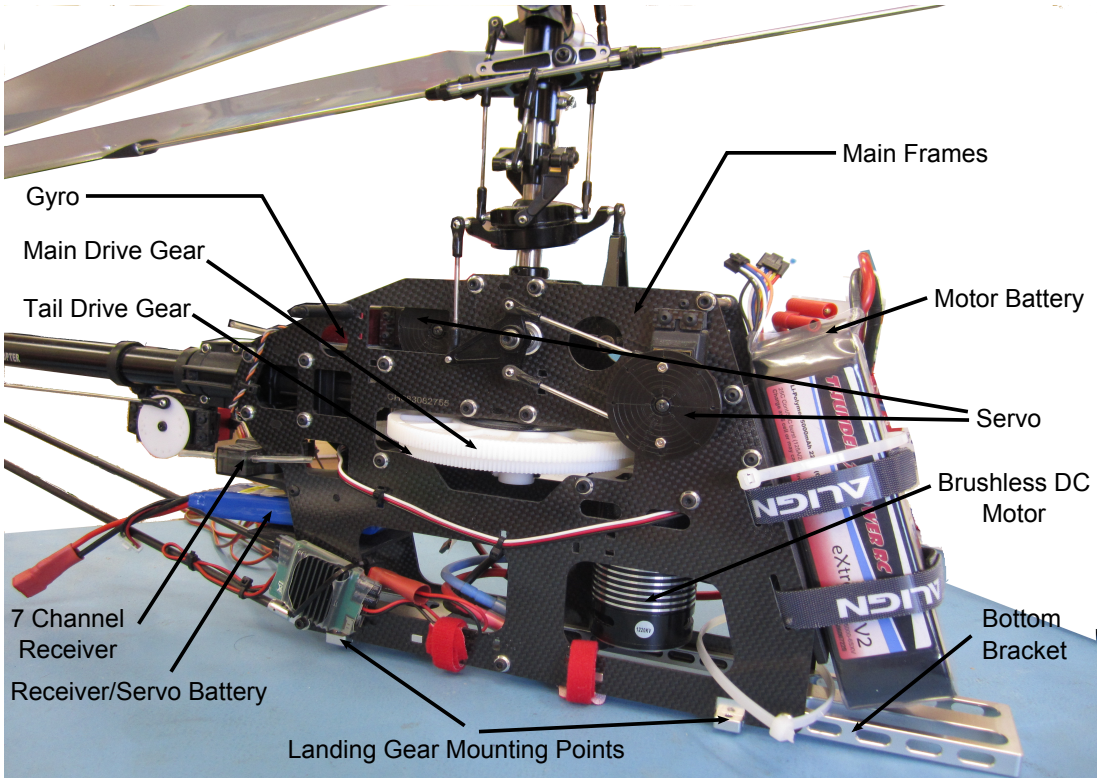


Figure 3.2 Side frame system and components attachment in TREX600 main structure.

regarded as a secondary rotor attached to the shaft either below or above the main rotor position by an unrestrained teetering hinge. The stabiliser bar blade consists of two simple paddles which are attached to a rigid rod. The stabiliser bar receives the same cyclic pitch and roll inputs from the swash-plate but no collective input. The stabiliser mechanism introduces stability to the helicopter dynamics through the use of the gyroscopic effect of the stabiliser bar tip weights and the aerodynamic effect on the stabiliser bar paddles [Kim and Tilbury, 2004, Shim, 2000]. When the stabiliser bar rotates, the bar earns gyroscopic effect and it tends to remain in the same plane of rotation. This would make the helicopter momentarily maintain the current roll and pitch angle for a substantial time. Considering the helicopter model in Figure 3.3, in a hovering condition, the stabiliser bar angle β is known to be zero (level). If a wind gust or other disturbance knocks the helicopter out of its equilibrium state, the stabiliser bar will continue to rotate in the same inertial plane [Kim and Tilbury, 2004]. This would subsequently help the helicopter back to equilibrium state through stabiliser bar action on the cyclic angle of the main blade.

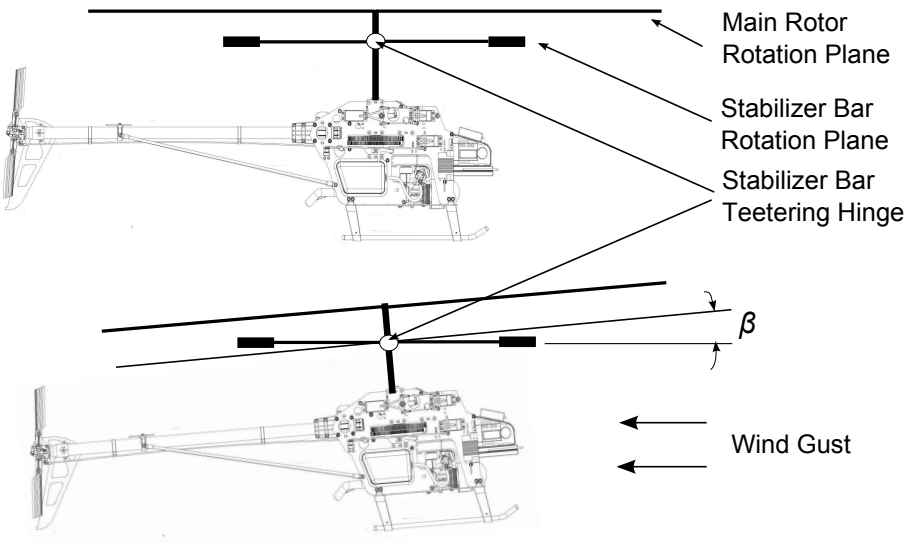


Figure 3.3 The stabilising effect of the stabiliser bar. Figure adapted from Kim and Tilbury [2004].

The TREX600 class helicopter uses a standard Bell-Hiller actuation system to control the angle of the main rotor blade. In contrast to fixed wing aircraft propeller, the blade angle of a helicopter's main rotor blade varies as it moves around the rotation plane in each cycle. The blade angle may be at a maximum at one point in its cycle around the mast and fall to a minimum value 180° later in the cycle. The main rotor blade angle is controlled through the action of the swash-plate. The swash-plate is a pair of plates, one rotating and the other fixed, positioned surrounding the rotor shaft that can be tilted in any direction by the control actuators as shown in Figure 3.4. Figure 3.4 shows two linkage arms coming from the swash-plate denoted as the Bell input and the Hiller input. The tilting motion of the swash-plate is accomplished by two cyclic actuators indicated as pitch servo input and roll servo input. The swash-plate is tilted fore and aft by the pitch servo input, and left and right by the roll servo input. The blade angle of the main rotor blades is manipulated by the combination of the Bell input from the swash-plate and the Hiller input from the stabiliser bar. The collective pitch input to the main rotor blade is achieved by lowering or raising the swash-plate. The washout arm with the slider prevents the collective input from affecting the stabiliser bar. The slider slides only when the collective input is applied and it remains stationary if only a cyclic input is applied.

If the swash-plate is tilted forward, for example, the blade angle of the main rotor

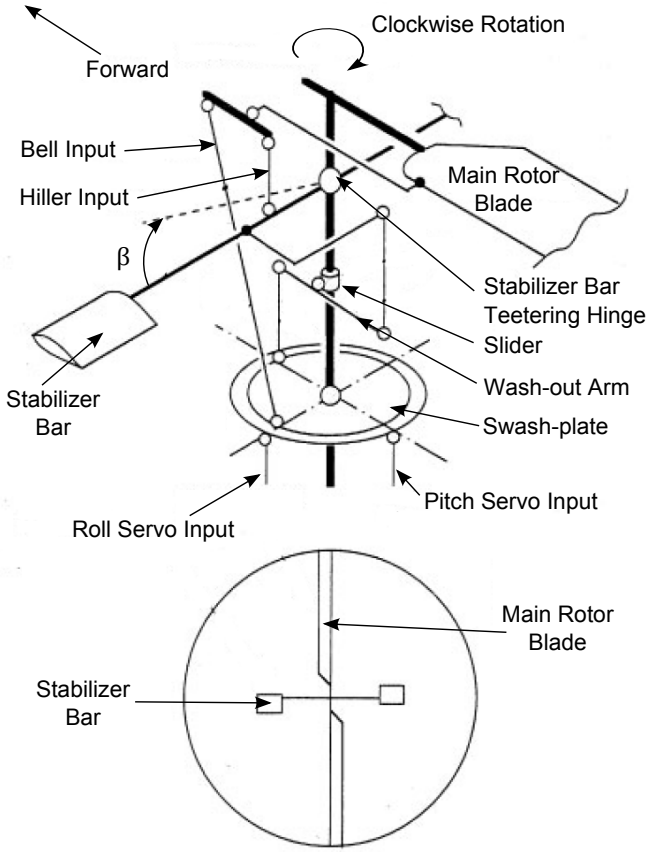


Figure 3.4 The basic cyclic and collective pitch mechanism in the helicopter control system. β indicates the stabiliser bar flapping with respect to the body coordinate frame attached to the rotor hub. Symbol '$\circ$' indicates ball joints and fixed joints are shown as '$\bullet$'. Figure adapted from Kim and Tilbury [2004].

blades is manipulated so that more lift is produced at the aft rather than the front of the rotor head. This tends to tilt the vehicle forward creating a forward component of force from the main rotor lift vector and thus creating forward motion. Similarly, a tilt of the swash-plate to the left causes more lift to be produced on the right than the left sides of the rotor disc. This asymmetry tilts the vehicle and moves it to the left. A tilt in any intermediate direction creates motion in that particular direction. The collective actuator moves the swash-plate up and down the rotor shaft and does not induce any tilt to the swash-plate. As the swash-plate moves up, the blade angle is increased by the same amount at all points through the cycle. This creates a uniform increase in lift across the disc with an overall increase in lifting force. In the TREX600 class helicopter, the motion of the collective actuator is achieved with a combination of three servomotors (120° Cyclic/Collective Pitch Mixing type) to push the swash-plate

up and down along the main rotor shaft.

A total of three electrically powered Futaba S9252 and one JR 8900G actuators or control servos are used to position collective, lateral cyclic, longitudinal cyclic and tail rotor linkages. The rotation speed of the brushless DC motor is controlled through an electronic speed controller (ESC). A small rechargeable 4.8 V Type C 2000 mA h battery provides power to the actuators through a 2.4 GHz AR7000 Remote Control (RC) receiver located in the servo mounting frame. The receiver processes signals transmitted by a hand-held Spektrum DX7 transmitter on the ground and produces Pulse Width Modulated (PWM) output signals to drive the servos. The AR7000 receiver combines an internal and external receiver, which offers superior performance in comparison with conventional narrow band systems. The Spektrum radio system simultaneously transmits two frequencies which create dual RF paths, and this virtually makes the system immune to internal and external radio interference. The receiver may accommodate as many as 7 different actuators where the channel assignments are shown in Table 3.2.

Table 3.2 RC receiver output channels

Receiver Channel	Output	Description
CH1	THRO	Electric motor/Electronic speed control (ESC)
CH2	AILE	Collective/Cyclic pitch movement
CH3	ELEV	Collective/Cyclic pitch movement
CH4	RUDD	Tail rotor cyclic pitch movement
CH5	GEAR	Auxiliary Channel
CH6	AUX1	Collective/Cyclic pitch movement
CH7	AUX2	Rate Gyro Sensitivity Switching
CH8	BATT	Power Supply (5V)

For the throttle, collective, lateral cyclic and longitudinal cyclic pitch, the receiver outputs are routed directly to the actuators/servos as shown in Figure 3.5. However, for the tail rotor cyclic pitch, the receiver output command is routed to a JR Angular Vector Control System (AVCS) G770 3D piezoelectric rate gyro which serves as a yaw damper. The yaw damper senses angular turn rate about the z-axis and uses this information to stabilise the helicopter in yaw motion. This feature allows the RC pilot to have better yaw control as all helicopters experience considerable cross-coupling

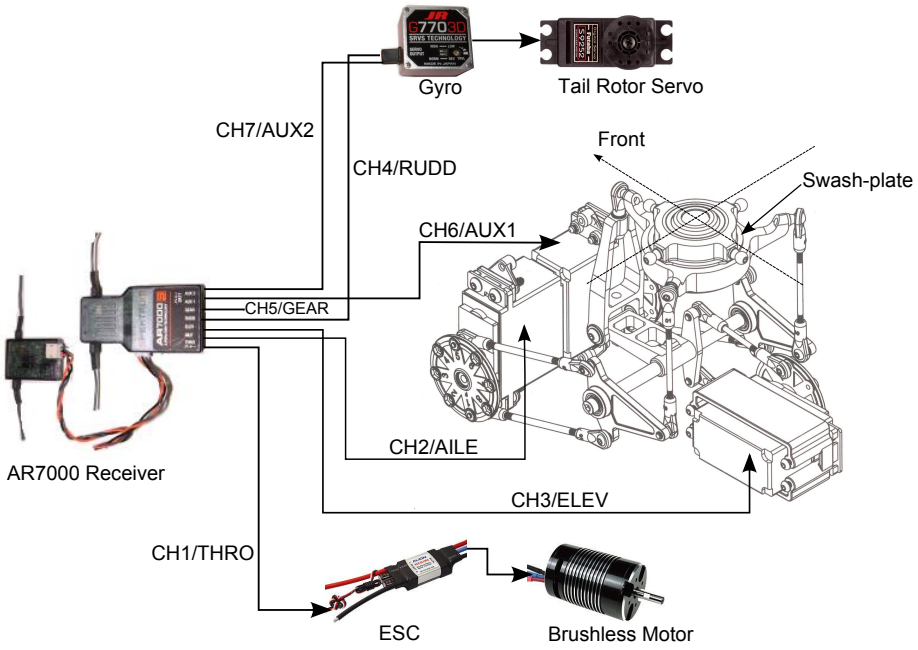


Figure 3.5 Spektrum AR7000 receiver and actuator (servo) connections.

between control inputs. For example, as collective is increased and the main rotor produces more lift, it also produces more torque for the tail rotor to counteract. The yaw damper senses the yaw created and sends a countering signal to the actuator, even if no input is commanded by the RC pilot. JR AVCS rate gyro is also useful when helicopter encounters a crosswind while hovering. When the helicopter drifts, the gyro generates a control signal to stop the drift and at the same time computes the drift angle and constantly outputs a control signal that resists the crosswind. Therefore the drifting of the tail can be stopped even if the crosswind continues to affect the helicopter position (see Figure 3.6).

The Spektrum DX7 transmitter contains two primary levers for vehicle control. The up-down movement of the right lever controls the throttle and collective pitch while left-right movement control the lateral cyclic. Left-right motion of the left lever of the transmitter controls the tail rotor, while up-down motion simultaneously controls the longitudinal cyclic. The Spektrum DX7 transmitter features mixing capabilities that adjust the tail rotor to roughly compensate for the change in torque created as the throttle and collective pitch increase. The DX7 transmitter also has the capability that

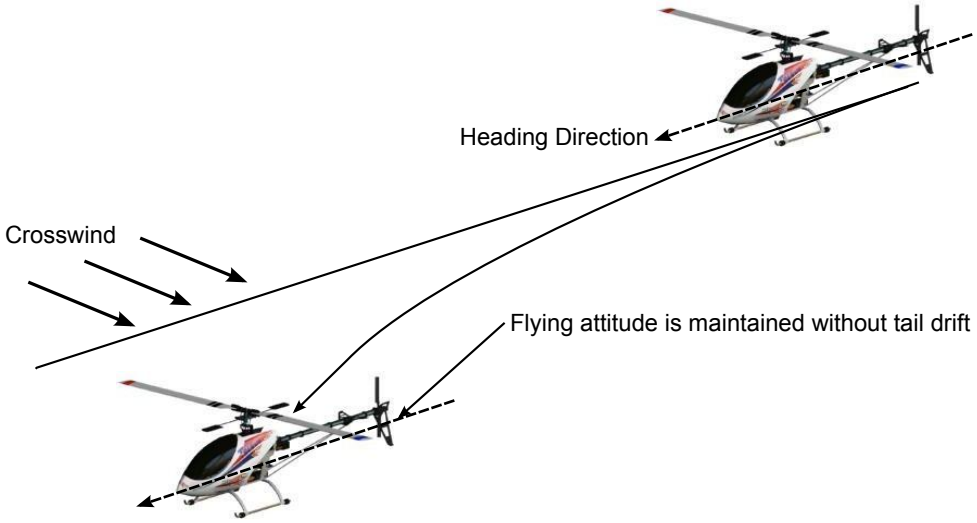


Figure 3.6 The gyro automatically corrects changes in the helicopter tail trim by crosswind.

allows us to connect any two DX7 series transmitter together for the purpose of training a new pilot. In practice, one of the two transmitters is held by the instructor pilot (Master Transmitter) and the second transmitter serves as the Trainer Transmitter. As long as the instructor holds his trainer switch in the 'ON' position, the model will respond to the commands of the trainer transmitter allowing the trainee to fly the model. The Master Transmitter has full control of the model if the trainer switch on the instructor's transmitter is left in its 'OFF' position.

3.3 THE DEVELOPMENT OF TEST STAND FOR SAFE FLIGHT CONTROL TESTING

Controlling a remote control helicopter is difficult and careful experimentation is essential in building a working UAV helicopter prototype. For the testing of the autopilot system developed in this research, a six degree of freedom (DOF) test-bed for safe indoor helicopter flight was developed. It was mainly designed as a safety device for preventing crashes and out-of-control flight. There are already many test stands for helicopter control tests available in the literature, but most of these stands are limited to only a few DOF that cannot be used to simulate free flight of the UAV while on the test stand [Cetto et al., 2009, Amidi, 1996, Kim and Tilbury, 2004, Vitzilaios and Tsourveloudis, 2009, Vilchis et al., 2003]. Therefore there is a need for a development of a test rig that

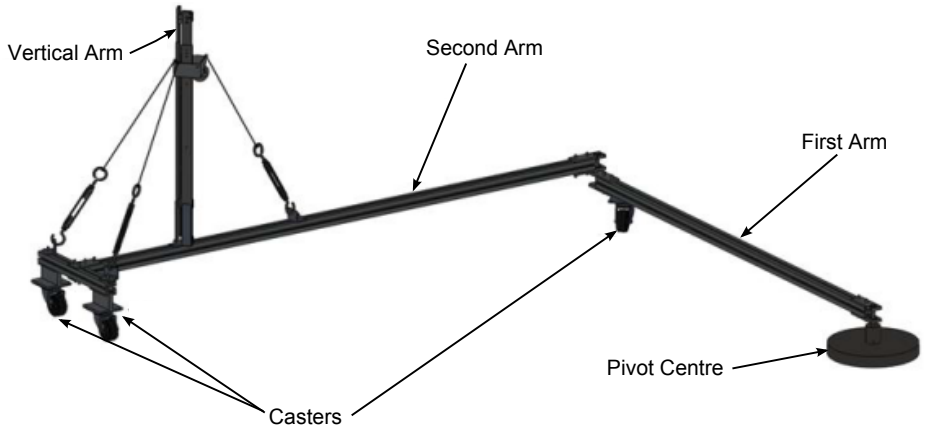


Figure 3.7 Translational functionality of the test stand.

allows the helicopter to execute a full 6 DOF movement with limited work space. The safety test rig system could also be equipped with instrumentation sensors that provide position and orientation of the helicopter UAV. Such a system can provide the ability to test the controller continuously regardless of the weather conditions. Subsequently, the development of such device would also minimise the need for experienced helicopter pilots while conducting flight tests.

3.3.1 Design Concept of The Test Stand

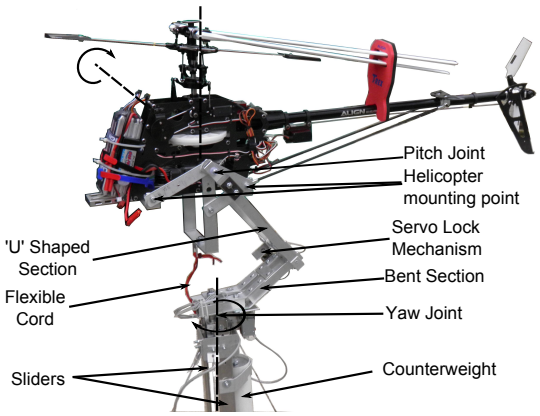
The final design of the test stand consists of two sections, one providing translation motion and while the other introducing rotational motion to the stand. Translation in the horizontal plane is achieved through the use of two jointed Selective Compliant Articulated Robot Arms (SCARA). The design offers faster motion response and lighter design than a comparable Cartesian system. Figure 3.7 shows the translational motion functionality of the designed test stand. The arms are mounted on a set of caster wheels which support the weight of the test stand. The first arm pivots off an anchor (a 15 kg steel weight) using deep groove ball bearings. The second arm pivots off the first arm, but it is longer than the first arm, and the helicopter is mounted such that it passes directly over the central pivot.

Vertical translation motion is achieved using linear bearings, made from two drawer sliders, mounted perpendicular to each other. There is 500 mm of vertical travel. The

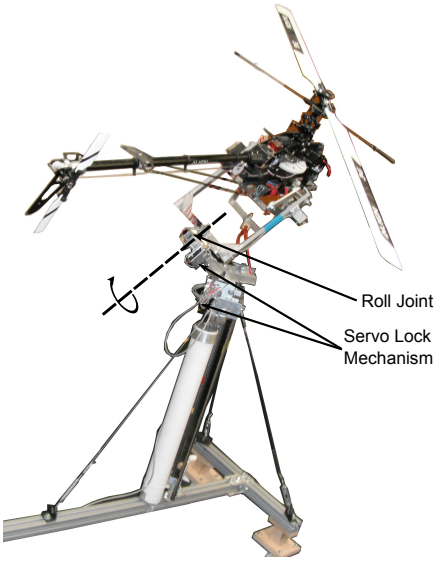
weights of the sliders, and of the pitch, roll and yaw joints, which are mounted at the top of them, are cancelled out by a counterweight. This means that the helicopter only has to lift its own weight, as if it was in free flight without the test stand. There are three stays which are attached to the top of the static part of the sliders, and to three points on the second horizontal arm. This gives rigidity to the sliders with minimal weight gain.

The helicopter platform is joined to the test stand by the mount shown in Figure 3.8. Rotational motion is achieved by the pitch, roll and yaw joints in the test frame. This arrangement allows the helicopter to have a variable tilt around pitch, roll and yaw axis. In order to limit the roll and pitch rotation movements, the helicopter motion is constrained by a flexible cord joining the bottom of the helicopter mount to the base of the yaw arm. The cord prevents the combined angle of the pitch and roll from exceeding the maximum tilt angle. By altering the length of the cord the maximum tilt angle can be restricted to a smaller angle. The maximum tilt of the roll and pitch movements are set about 30° around pitch and roll axis. The yaw motion is constrained to approximately 450° due to cable twist issues. The pitch, roll and yaw arrangement consists of two sections; a 'U' shaped section and a bent section. Pitching motion occurs by a rotation between the helicopter mount and 'U' shaped section. The yaw motion occurs by the rotation between the bent section of the helicopter occurs by coupled rotation of the 'U' shaped to bent section and the bent section to the vertical member.

The rotational joints (pitch, roll and yaw) can also be locked into to zero position with three solenoids. This is necessary for the zeroing of the instrumentation before testing and during the wind up phase of the helicopters rotors. During wind up phase, the rotor blades can be unstable and behave chaotically. The helicopter needs to be held in a steady position until steady rotation speed of the blades has been reached. The metal shaft of the solenoids is released via locking switch allowing the helicopter to tilt and yaw. In the first design of the test frame prototype, the solenoid shafts are returned back to their respective initial position by using elastic bands. Since the solenoids only allow single direction of movement, re-engagement process of the solenoids to lock the pitch, roll and yaw motion must be done manually by aligning the solenoid pins with



(a)



(b)

Figure 3.8 The roll, pitch and yaw motion arrangement in test stand. (a) Test stand view from the side; and (b) Test stand view from the back.

their targets before flipping the locking switch.

The returning mechanism of locking metal shafts is further improved by replacing the solenoid mechanism with rotational servomechanism. Figure 3.9 shows the operation of the servomechanism to lock the rotation motion. Simple servo controller was used to control the movement of the servos. The rotational movement of the servos can be translated into the linear motion of the metal shaft by using a small scale SCARA arm.

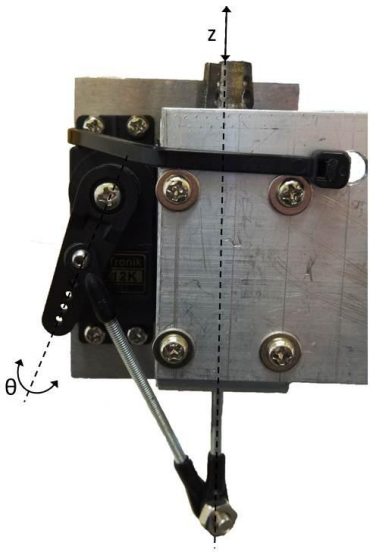


Figure 3.9 The servo locking mechanism of the rotational motion (roll, pitch and yaw) of the test rig.

3.4 FLIGHT INSTRUMENTATION SETUP FOR AUTOMATIC FLIGHT CONTROL TEST

The overall architecture for an automatic flight control system consists of components such as the remote controlled (RC) helicopter platform, the flying test stand, an on-board computer systems and a ground control station (GCS) as shown in Figure 3.10. Generally, the automatic flight control system is divided into ground and on-board components. The altitude (linear position transducer) and translational position (rotary encoders) sensors are mounted on the test stand and wire connected to the on-board computer's digital input-output (DIO) port. The on-board computer system is considered the most important part in the overall system. It functions as a data logger in charge of collecting necessary flight data from sensors and servo inputs, handling switching between autonomous control and manual control as well as implementing flight control laws. The manual control system, which is normally a radio controlled joystick transmitter, is used by a human pilot to control the helicopter in manual flight tests. The standard RC system serves as a backup in case the autopilot fails and it is also very useful in the process of controller design. Lastly, the GCS system is used as on-line monitoring of the UAV helicopter and to communicate with the on-board computer system to control some logical states and controller settings from ground.

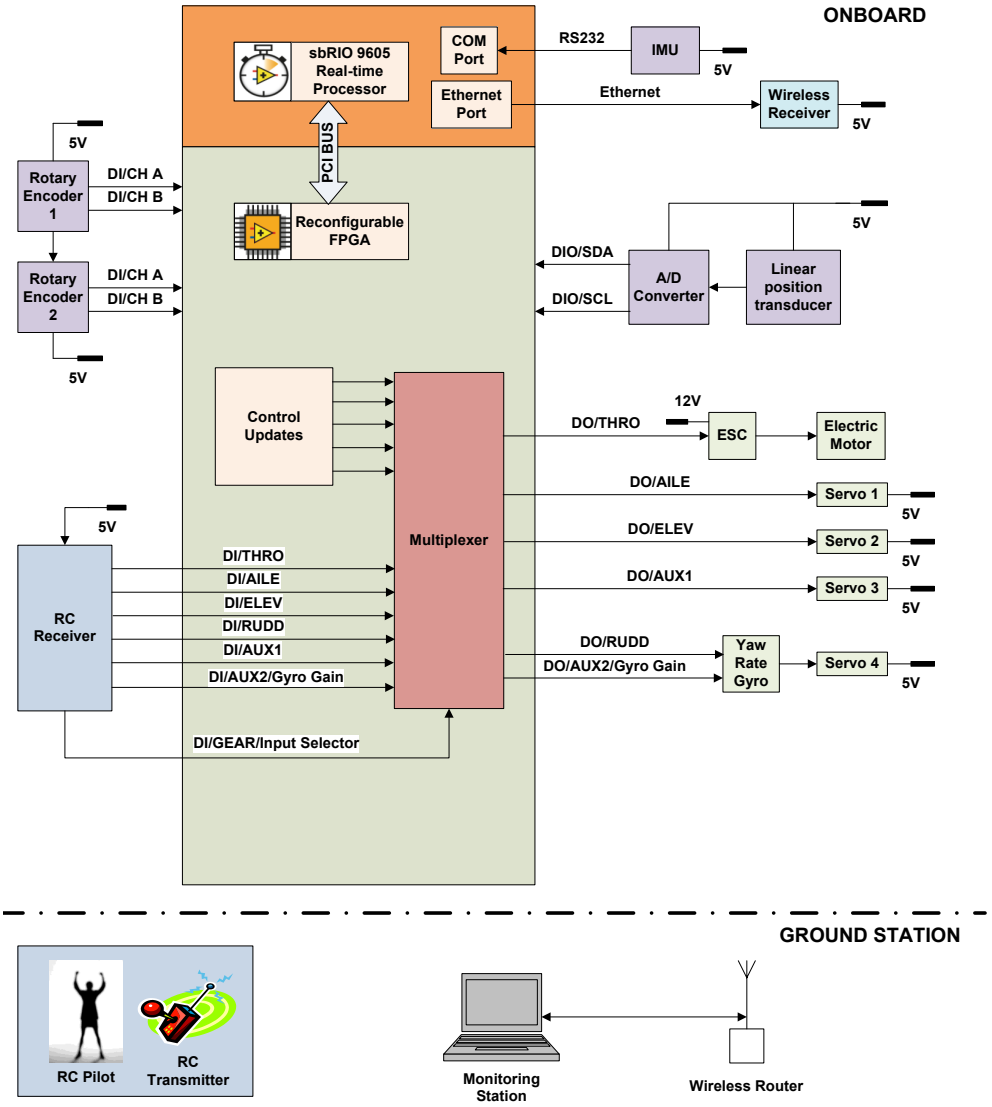


Figure 3.10 The overall architecture for the developed automatic flight controller system.

The communication between the on-board computer and the GCS is done via wireless network.

3.4.1 Positioning and Orientation System

For indoor testing of the automatic flight control system, the safety test rig had been equipped with several rotary encoders and a linear position transducer to give us sensory information about the position of the helicopter UAV. Furthermore, an inertial measurement unit (IMU) was also installed on the helicopter to obtain attitude information of the aerial system. The motivation to use the linear position transducer and

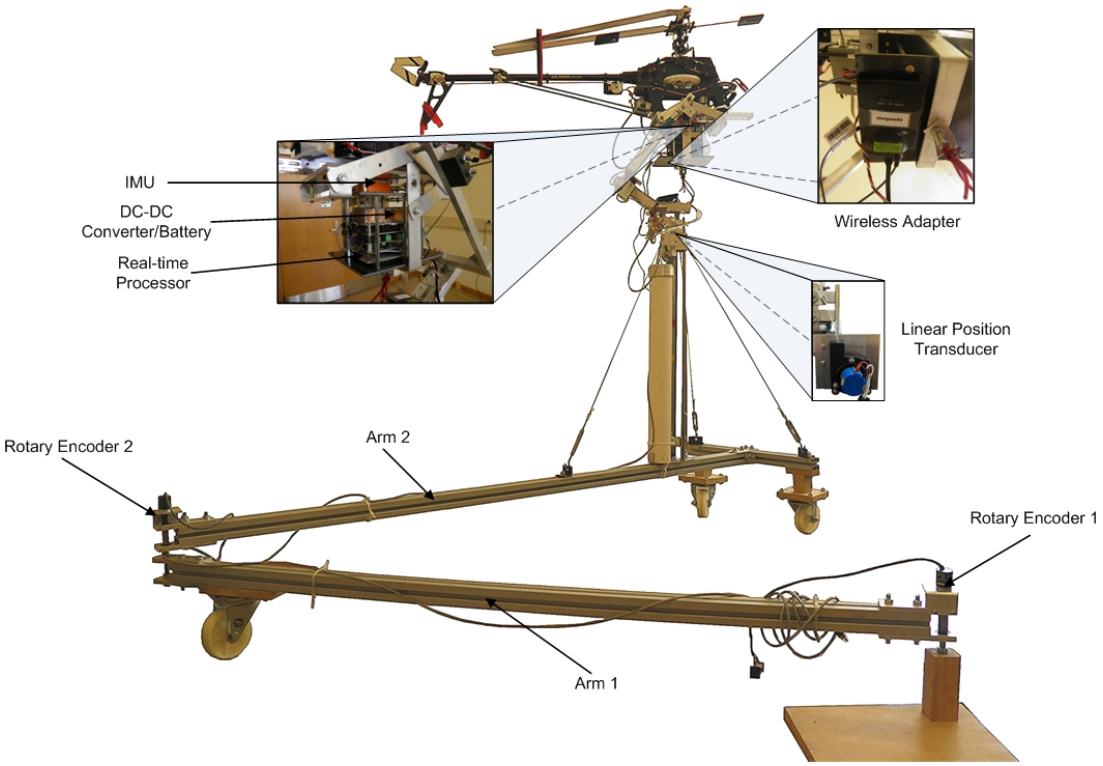


Figure 3.11 Helicopter avionics and sensors on the test stand.

rotary encoders instead of vision based localisation system was to avoid high development cost incurred from the use of high cost cameras [Vitzilaios and Tsourveloudis, 2009, Valenti et al., 2006, 2007]. Figure 3.11 shows the arrangement of positioning and rotational sensors in the test stand. Two rotary encoders are attached at each joint of arm 1 and 2 in the test stand. These encoders are set to zero during the initialisation phase and produce signed numbers that indicate the current position relative to the initial position. A positive number gives a rotation reading that indicates movement to the left, and vice versa, a negative reading indicates movement to the right.

Measuring the rotational displacement of the joints between the centre pivot and the first arm (θ_1), and the first and second arm (θ_2), gives the location of the helicopter in the XY plane. To calculate the XY coordinate of the helicopter relative to the base and initial position, the following equation is used:

$$x = L_1 cos\,(\theta_1) + L_2 cos\,(\theta_1 + \theta_2) \tag{3.1}$$

$$y = L_1 sin\,(\theta_1) + L_2 sin\,(\theta_1 + \theta_2) \tag{3.2}$$

where L_1 and L_2 indicate the length of arm 1 and 2 respectively.

Due to cost constraints, it was decided that positional measurements should be accurate to 1%. To achieve this accuracy in the XY plane, the encoders must have a resolution finer than 0.57° ($\theta = \sin^{-1}(1\%)$). The Romuron A6A2-CWZ3E incremental optical encoder was selected for this project, with sufficient accuracy ($\theta = 360°/2000 = 0.18°$ revolution) for a prototype. The encoder is compact and has a very small starting torque, so it will easily fit into confined spaces and will not add any dynamic effects to the test stand.

In order to measure the altitude of the helicopter, a linear position transducer is used to monitor the actual altitude reading of the helicopter. The linear position transducer is mounted at the side of the sliding vertical mechanism as shown in the Figure 3.11. The UniMeasure LX-PA-30 linear position transducer was chosen to measure the displacement of the sliding vertical arm. The unit consists of a precision potentiometer, attached to a spring loaded string (maximum extension = 750 mm), which unwinds as the vertical arm is extended. It has linearity of ±0.25% which produce distance measurement accuracy up to 1.875 mm. There are also dead band regions at the maximum and minimum allowable displacements which need to be accounted for in the string length displacement, l calculation. The linear transducer outputs an analog signal measurement and this signal is converted to digital signal with 12-Bit MCP3221 A/D converter chip with I^2C interface. The signal received from the A/D converter chip is used to calculate the length of the string, l by the following relationship:

$$\begin{aligned} V_{in} &= V_{dd} \times \frac{A_{in}}{2^{12}} \\ l\,(\text{mm}) &= \frac{V_{in} - V_{o,low}}{V_{dd} - V_{o,low} - V_{o,high}} \times 750 \end{aligned} \tag{3.3}$$

where A_{in} and V_{dd} denote output voltage and excitation voltage supplied to the potentiometer. $V_{o,low}$ and $V_{o,high}$ indicate the offset voltages of the potentiometer corresponding to the lowest (initial) and highest (maximum) string displacement. Since the linear transducer is not placed parallel with the vertical arm, a simple modification is further made into the altitude measurement from the linear transducer. Figure 3.12

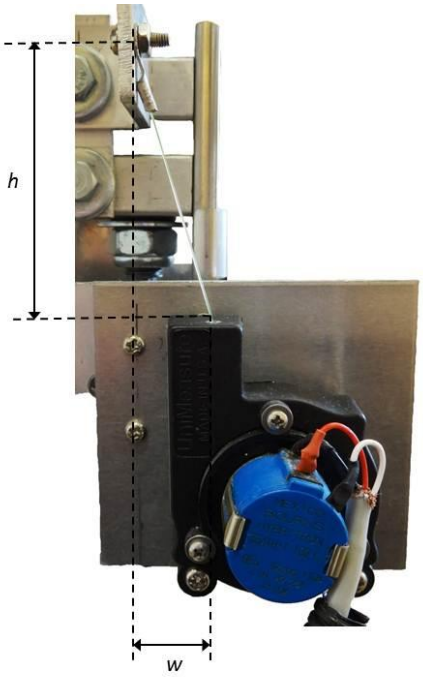


Figure 3.12 The close up view of the linear UniMeasure LX-PA-30 position transducer.

shows the placement of linear transducer on the safety test rig. Notice that there is about $w = 13$ mm horizontal distance displacement between the string attachment on the test rig and the mount of the sensor. By using simple Pythagoras theorem, the altitude of the helicopter is calculated as follows:

$$h\,(\text{mm}) = \sqrt{l^2 - 13^2} \tag{3.4}$$

3.4.2 On-board Controller

The flight computer system used in this project is based on the National Instrument (NI) Single-Board RIO device (NI sbRIO-9605). The NI sbRIO-9605 is an embedded device that combines a real-time processor, reconfigurable field-programmable gate array (FPGA), and digital I/O on a single circuit board, programmed with NI LabVIEW® software. It features a fast 400 MHz processor, a Xilinx Spartan-6 LX25 FPGA, and a high-speed and bandwidth connector card that provides direct access to the processor's 96 3.3 V digital I/O FPGA lines. It also provides 128 MB of DRAM for embedded

operation and 256 MB of non-volatile memory for storing programs and data logging. This device features a built-in 10/100 Mbit/s Ethernet port that can be use to conduct communication over the network. For sbRIO-9605 variant, only a RS232 serial port is provided to control peripheral devices.

The main tasks of the real-time processor in the NI sbRIO-9605 are to gather sensory information from the IMU and to compute the necessary control update. On the other hand, the FPGA in the on-board computer system is programmed with Labview® software to functions as a data logger in charge of collecting positioning data from linear transducer, rotary encoders and servo inputs. It is also responsible for handling switching between autonomous control and manual control as well as updating the servo actuators according to the computed flight control laws. During manual flight mode, the helicopter is controlled remotely using standard RC equipment by human pilot. While in the autonomous flight mode, the helicopter movement will be directly controlled and supervised by the on-board flight controller.

3.4.3 Inertial Measurement Unit

The IMU sensor is used to measure the orientation of the helicopter UAV in the body frame reference system. Attitude determination of a UAV is critical in order to ensure maintained flight for the autopilot system developed in this project. The Xsens MTi IMU was selected for UAV attitude measurement. It produces measurements of 16 calibrated states consisting of Euler angle (ϕ, θ, ψ), in quaternion representation (q_0, q_1, q_2, q_3), angular rates in body coordinate frame (roll rate, p, pitch rate, q and yaw rate, r), body accelerations (A_x, A_y, A_z) and magnetic fields (m_x, m_y, m_z). The orientation of the MTi is computed by a Kalman filter algorithm for 3 DOF orientations. The Kalman filter algorithm uses signals from the rate gyroscopes, accelerometers and magnetometers to compute an optimal 3D orientation estimate of high accuracy with no drift for both static and dynamic movements. Figure 3.11 shows the location of this sensor on the helicopter. The MTi IMU outputs ASCII data at 100 Hz via RS232 protocol at 115 200 bit/s with no flow control.

The IMU provides Euler angles representation ranging from $+\pi$ rad to $-\pi$ rad.

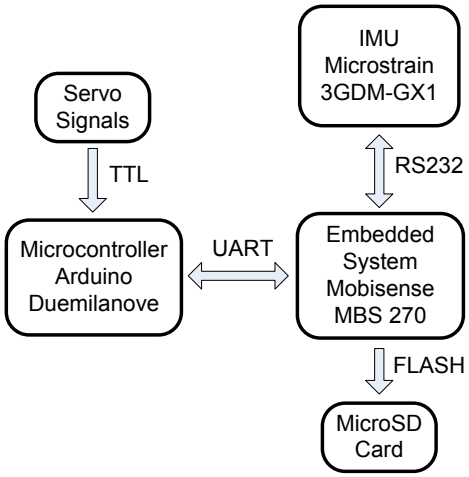


Figure 3.13 Overview of the measurement system setup.

The Euler angles representation suffer from the discontinuity problem when the body rotation angle rotates beyond 180° (or −180°). This problem should not happen in roll and pitch axis as the helicopter UAS is not required to do inverted flight or aerobatic manoeuvres. However, this does not hold in the yaw rotation axis since the yaw angle has unlimited rotation. Modification to yaw angle measurement should be done in the software to eliminate the discontinuity problem in yaw axis at each sampling point. This can done through the use of geometry function in Labview®.

3.5 FLIGHT INSTRUMENTATION SETUP FOR SYSTEM IDENTIFICATION

System development for this project was implemented in different stages which results in different instrument setup for both automatic flight control test and system identification test. The main purpose of system identification is to model or predict the helicopter response based on the collected flight data. In order to test the proposed system identification method, a much simpler data acquisition system is designed to record the necessary data during flight. The overall architecture overview of the data acquisition, shown in Figure 3.13, consists of a Mobisense MBS270 embedded computer, a Microstrain 3DM-GX1 Inertial Measurement Unit (IMU) and an Arduino Duemilanove microcontroller.

The Mobisense MBS270 embedded computer was basically used as the central processing board, primarily to collect and store all sensory data on the MicroSD card. The small computer board was equipped with a MicroSD storage card slot, serial interface for communication with the IMU and general purpose inputs/outputs for measurement and actuation purposes. Furthermore, the board offers faster software development since it runs on Linux OS, enabling higher-level programming in C/C++ with the supplied libraries. The board features the MMX instruction function set and Direct Memory Access for video processing which makes the board as an excellent platform for video processing tasks.

The IMU unit which measures the linear acceleration, angular rates and Euler-angles of the helicopter was directly connected to the embedded system through an RS232 serial interface. This module can be programmed to run in polled mode (measurements start upon request) or in continuous mode. In continuous mode, the IMU transmits measurements to the host in the requested output format at the highest possible rate. For system identification purposes, it is recommended to use the *Send Gyro-Stabilised Quaternion and Vectors* (0×0C) output format in continuous mode. This would give noise free accelerations and quaternion readings that provide unique measurements for the attitude and the bias corrected angular rate vector. The highest possible output rate in this mode is 100 Hz, which can be achieved by modifying the default settings stored in the EEPROM address.

The four pilot stick positions on the radio transmitter are the inputs in the system identification. During the experiment, the servo signals received from the transmitter signals were measured instead of direct measurements of the pilot stick deflections. There are four servomotors in the TREX600 actuation system, one of which controls the yawing movement of the helicopter. The remaining three servos which are arranged at 120° around the swash plate are used to apply a combination or individual efforts of lateral cyclic, longitudinal cyclic and collective pitch input movements. During the experiments, a separate micro-controller was used to continuously measure the servo signals. It passes the values through to the MBS270 over the UART serial interface.

3.6 SUMMARY

In this chapter, the air vehicle platform used for the system identification and control algorithm testing is discussed in terms of the vehicle specifications, structural, stability augmentation and actuation system characteristics. In order to ensure that flight controller tests were conducted in safe manner, a 6 DOF safety test stand was developed and equipped with necessary instrumentations that provide the end user with the helicopter's position and attitude information. The benefit of having such a test system is that it would provide us with the ability to test the controller continuously regardless of the weather conditions. Subsequently, the development of such a device would also minimise the need for an experienced helicopter pilot while conducting flight tests. The system overview of flight instrumentation and avionics system used on the test stand is also discussed. At the end, the system overview of flight instrumentation developed for collecting data during system identification test is presented. The theoretical foundation and system identification methodology using neural networks approach are presented in the next chapter.

Chapter 4

NEURAL NETWORK BASED SYSTEM IDENTIFICATION

4.1 INTRODUCTION

In this chapter, the fundamental theories of neural network (NN) architectures, training algorithms and NN based system identification are presented. The system identification method enables us to infer a representation of the dynamic model based on input and output data observed from the system. Many conventional system identification methods exist to model the dynamic system such as methods based on least square estimations maximum likelihood estimation and Kalman filtering as reported in Grauer et al. [2009], Samal [2009], Chowdhary and Jategaonkar [2010]. Neural network based system identification is an alternative method that has been proven as an efficient tool for identification of complex and non-linear systems without detailed analytical descriptions of the system. It is particularly useful for certain applications where it is hard to model the dynamics of the system using the first principle modelling approach that consists of fundamental laws of mechanics and aerodynamics analysis.

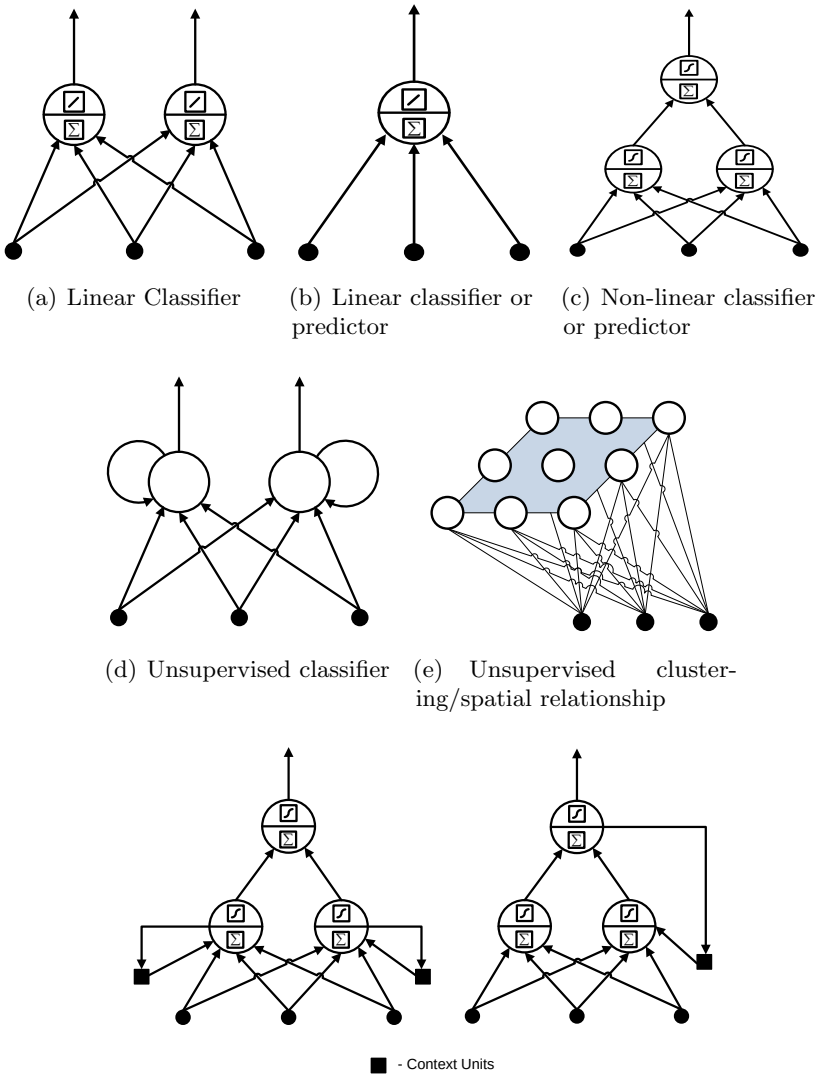
This chapter is organised as follows: Section 4.2 introduces the fundamental concept of artificial neural network (ANN). Several architectures of feed-forward NN such as Multi-layered Perceptron (MLP), Hybrid Multi-layered Perceptron (HMLP) and recurrent Elman network are described for system identification application. The NN based system identification methods are introduced in Section 4.3 for modelling the non-linear dynamics of the helicopter UAS considered in this project. In this section, flight data collection, model structure selection, NN training procedures and model

validation are presented. Finally, the chapter is summarised in Section 4.4.

4.2 THE ARTIFICIAL NEURAL NETWORKS

Artificial neural network (ANN) is a computational model consisting of a collection of interconnected neurons that have an ability to learn complex mapping (linear or non-linear) in the provided input-output data. It can also provide reliable predictions for new situations containing even noisy or partial information [Samarasinghe, 2007]. The mathematical computation of ANN has been largely constructed and inspired by the biological neurons in the human brain. Haykin [2009] states that the ANN resembles the human brain in two aspects: (1) the knowledge is gained by the networks through a learning process; and (2) the knowledge gained from the learning process is stored in the weighted interconnection between neurons or synaptic weights.

The ANN can perform a diverse range of tasks including prediction/forecasting, function approximation, pattern classification and clustering. Throughout the ANN literature, there are a variety of ANN architectures that have been developed to achieve the mentioned tasks. The typical ANN types are shown in Figure 4.1 [Samarasinghe, 2007]. A single layer perceptron in Figure 4.1(a) is regarded as a linear classifier similar to simple and multiple discriminant function analysis in the field of statistics. The linear neuron shown in Figure 4.1(b) can function either as a linear classifier or predictor. In its prediction mode, the neuron's predictive capabilities are similar to simple or multiple linear regression models in statistics. The Multi-Layered Perceptron (MLP) network shown in Figure 4.1(c) is the most popular neural network type which is suitable for non-linear classification and prediction tasks. The competitive network in Figure 4.1(d) is a network with unsupervised learning that is capable of finding clusters in the data. The self organising feature map (SOFM) competitive network shown in Figure 4.1(e) is also capable of finding unknown clusters in the data; and has the ability to preserve the spatial relationship of the data and clusters [Samarasinghe, 2007]. In Figure 4.1(f), two types of recurrent networks (Elman and Jordan network) are frequently used for time-series forecasting. These networks contain feedback links and memory/context units to capture the temporal data effects [Pham and Liu, 1993, Kalinli and Sagiroglu,



(a) Linear Classifier (b) Linear classifier or predictor (c) Non-linear classifier or predictor

(d) Unsupervised classifier (e) Unsupervised clustering/spatial relationship

(f) Time series forecasting

Figure 4.1 Different types of NN modelling architectures: (a) Single-layer perceptron; (b) Linear neuron; (c) Multi-layer perceptron (MLP); (d) Competitive network; (e) Self-organising feature map (SOFM); and (f) Recurrent networks

2006]. In this chapter, the MLP, Elman network and their respective variants are used for the system identification application.

All of these networks presented in Figure 4.1 contain many links connecting inputs to neurons and from neurons to outputs. These network connections are also popularly known as weights which are allowed to freely adapt to the training data by the learning algorithms. These weights are also regarded as free parameters as in regression modelling in statistics. Thus, this would make the neural networks similar to the parametric models involving the estimation of optimum parameters [Samarasinghe, 2007].

The computational model of a single neuron that can perform intelligent functions or tasks is shown in 4.2(a). As an example, the process to produce a pulse-width output signal inside a single neuron is shown in detail in Figure 4.2(b). The processing unit or neuron accepts external inputs through the weighted connections, W_{hj} and all of these signals are summed up before being fed to the activation function. External inputs such as data measurement or outputs from the other neuron's calculation can be supplied to the neuron for processing. If the activation function in the neuron is a linear function, this NN model is also known as the adaptive linear neuron model (ADALINE) which was first developed by Widrow and Hoff [1960]. The learning of the ADALINE network was done by minimising the square error between the target value and the output of the model. This learning process is also known as gradient descent in neural learning or least square error minimisation in statistical methods [Samarasinghe, 2007].

The activation function, f_h in the neurons is used to introduce non-linearity into the network. The activation function is typically selected as non-linear and continuous function that remains bounded within some upper and lower bounds [Norgaard, 2000, Samal, 2009]. Since the output of non-linear function varies non-linearly with the input, the NN model can perform non-linear mapping between inputs and outputs. Hornik et al. [1989], Tu [1996] and Tu [1996] further proved that a non-linear activation function such as sigmoid function introduced in the neurons would make the NN capable of approximating any non-linear function of interest to any desired degree of accuracy with sufficiently available neurons. Hornik et al. [1989] further implies that any failure of a function mapping by a multilayer network must arise from inadequate choice of weight parameters or an insufficient number of hidden nodes.

The versatility of a NN as a universal approximator is not only limited to sigmoid function. Stinchcombe and White [1989] proved in their work that NN with general class of non-linear function can also achieve universal approximation. Furthermore, the non-linear function mapping can also be approximated by NN with a bell shaped activations function in the hidden neuron unit [Baldi, 1990]. Different activation functions are used in neural networks training and some of the activation examples are given in Figure 4.3. The step and sign functions are activation function typically used for binary

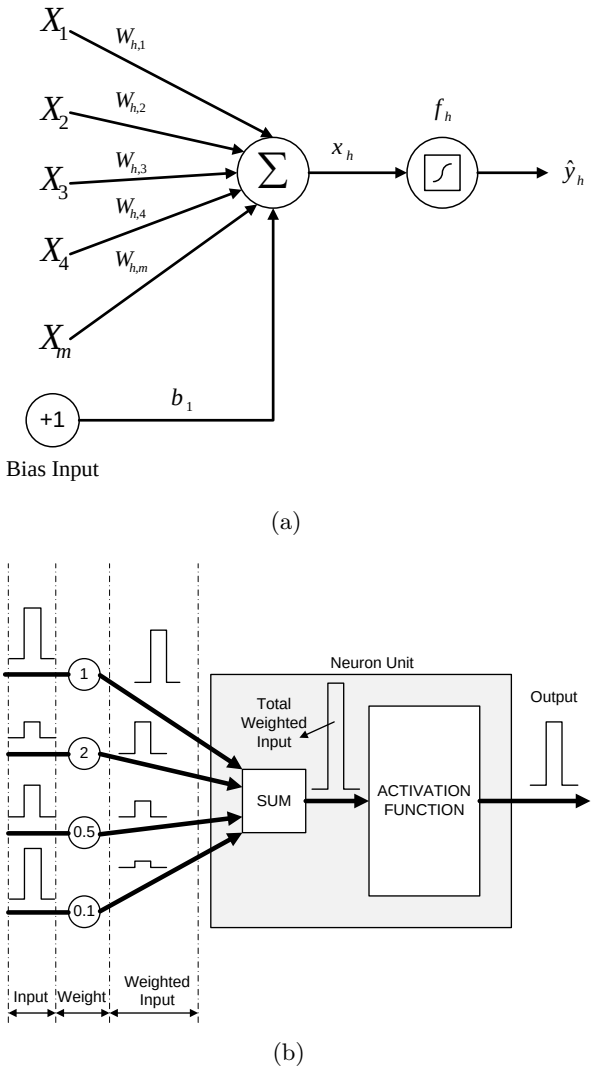
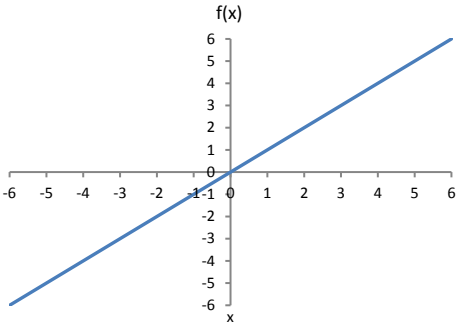


Figure 4.2 The artificial neuron model with multiple inputs and single output: (a) The output is represented in compact mathematical form as $\hat{y}_h = f_h\left(\sum_{j=1}^{m} W_{hj}X_j + b_1\right)$. The term h represents the number of neurons in the network and m is the number of inputs entering the neuron; and (b) The detailed working mechanism of a single neuron. Figure adapted from Samarasinghe [2007].

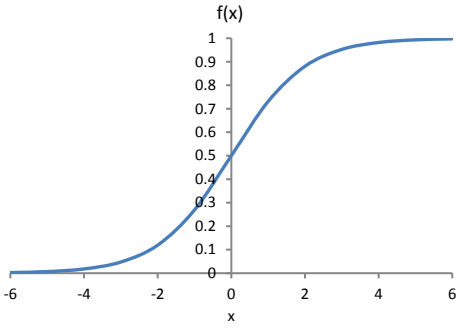
classification schemes. For system identification and regression modelling problems, it is common to use the hyperbolic tangent function in the hidden layer and linear function for output layer.

4.2.1 Multi-Layered Perceptron

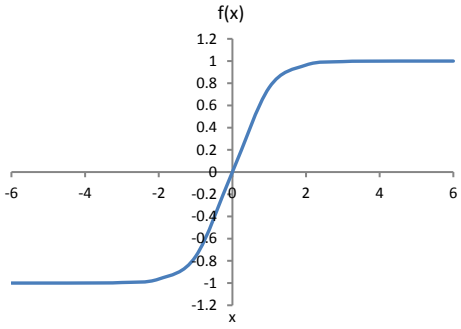
One of the most popularly used neural network architectures is based on feed-forward Multi-Layered Perceptron (MLP). Multilayer Perceptron (MLP) is a class of ANN which is built from several layers of neurons which only allows unidirectional signal



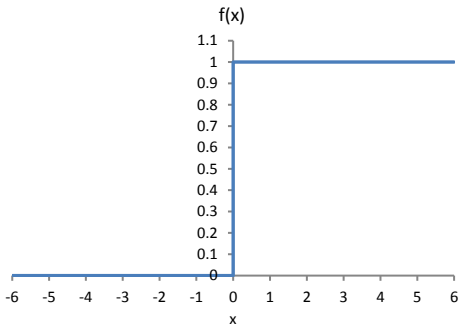
(a) $f(x) = x$



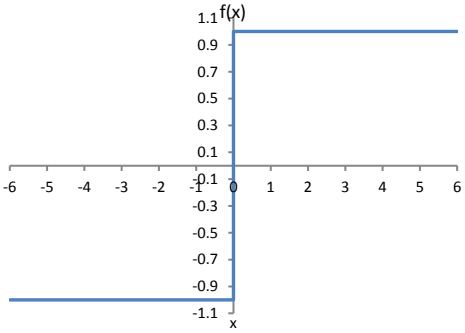
(b) $f(x) = \frac{1}{1+e^{-x}}$



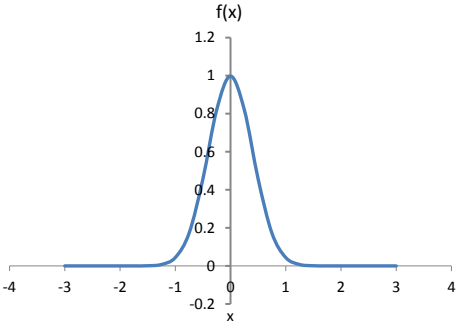
(c) $f(x) = \frac{e^{2x}-1}{e^{2x}+1}$



(d) $f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x < 0 \end{cases}$



(e) $f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$



(f) $f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$

Figure 4.3 Different types of activation function for NN modelling: (a) Linear function; (b) Sigmoid function; (c) Hyperbolic Tangent function; (d) Step; (e) Sign function; and, (f) Gaussian function with real constant $a, b, c > 0$

flow [Wilamowski, 2011b]. Typically, MLP networks are constructed with an input layer, one or more hidden layers and an output layer. These three layers are linked by weights connection resulting in two set of weights, the input-hidden layer weight and the hidden-output layer weight. The NN can be constructed with more hidden layers, however, there are no significant advantages or practical reasons in including

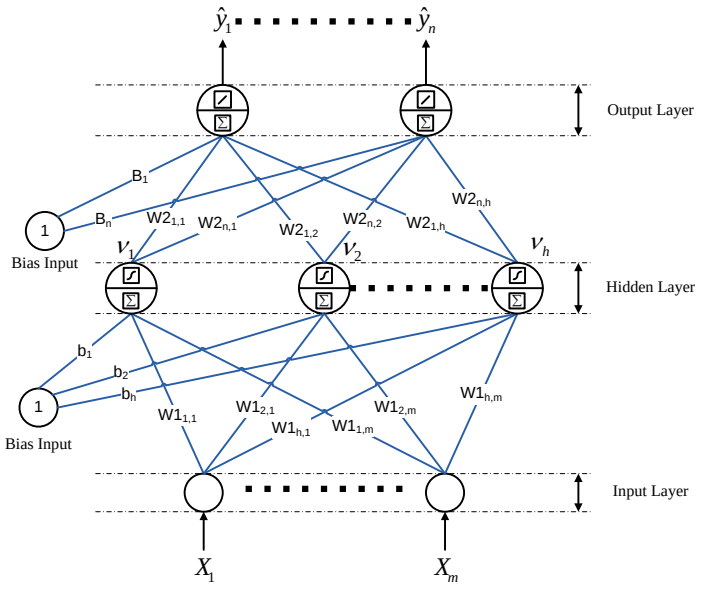


Figure 4.4 A fully connected, feed-forward multi-layered perceptron (MLP) with m-inputs, h-hidden neurons and n-outputs.

more hidden layers in regression problem [Tu, 1996]. Moreover, Funahashi [1989] and Cybenko [1989] have proven that any non-linear relationship function mapping can be sufficiently approximate with a single hidden layer with sigmoid activation function. The MLP architecture with one hidden layer is shown in Figure 4.4. For a single hidden layer case, the outputs formulation from hidden and output layer of a MLP network is given as follows:

$$v_h(t) = g_h\left(\sum_{j=1}^{m} W1_{hj} X_j(t) + b_h\right); \qquad \text{for } h = 1, 2, 3 \cdots H \tag{4.1}$$

$$\hat{y}_i(t) = g_i\left(\sum_{h=1}^{H} W2_{ih} V_h(t) + B_i\right); \qquad \text{for } i = 1, 2, 3 \cdots n \tag{4.2}$$

where $W1_{hj}$ is the weights matrix between the input layer and the hidden layer and $W2_{ih}$ is the weights matrix between the hidden layer and the output layer. The functions $g_h(*)$ and $g_i(*)$ are non-linear activation function for neurons in each hidden and output layers. The symbol H denotes the number of neurons in the hidden layer while b_h and B_i are the bias elements for the input layer and output layer. The number of inputs and outputs of neural network are presented by m and n respectively. In Equation (4.1) and (4.2), the weight connections and biases in the network structure are included in

parameter vector θ and are defined as:

$$\theta = [W1_{hj} \qquad W2_{ih} \qquad b_h \qquad B_i] \tag{4.3}$$

Here, the dimension of parameter vector θ is equal to the total number of weight connections in the network, d. The external inputs to the network are represented as time regression vector as follows:

$$\varphi\,(t) = [X_1 \qquad X_2 \qquad X_3 \ \cdots \ X_m] \tag{4.4}$$

The parameter vector θ is unknown and should be determined through the use of training algorithms that infer input and output relationship. A set of training data is normally presented to the training algorithms to obtain a suitable input and output relationship such that the different between measured output and prediction is below certain acceptable error threshold given as:

$$y_i(t) - \hat{y}_i(t) \leq e_i(t) \tag{4.5}$$

4.2.2 Hybrid Multilayer Perceptron

One aspect that we want to address in this work is to investigate whether the hybrid multi-layered perceptron (HMLP) network is more efficient than the standard Multilayer Perceptron (MLP) network. In this study, the HMLP architecture consisting of only one hidden layer is proposed to learn the non-linear relationship of the dynamics model. The HMLP network is a modified version of the popular Multilayer Perceptron (MLP) network.

In HMLP network architecture, the network contains extra connections that allow direct links between input nodes and output nodes of the network. This specific feature makes the HMLP different from MLP, because in the standard MLP structure, no connections are allowed to jump across hidden layers. The proposed HMLP network with one hidden layer is illustrated in Figure 4.5. Note that the HMLP possesses the same weight connections to hidden layer and output layer as in MLP except with some

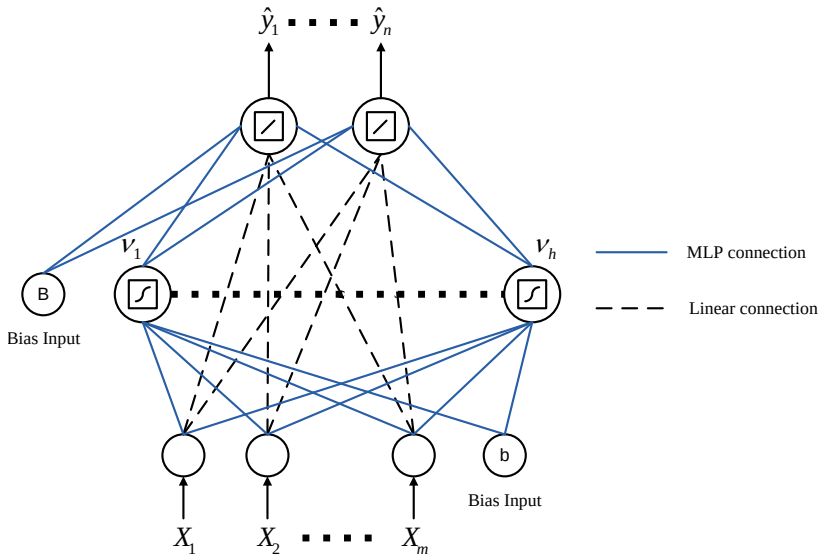


Figure 4.5 The hybrid multi-layered perceptron (HMLP). The dashed lines indicate extra linear connections in the network.

extra linear connections indicated in the figure. Mashor [2000] suggested that HMLP network structure offers improved training efficiency and generalisation properties for neural network modelling. Furthermore, the utilisation of linear connection in HMLP can significantly reduce the number of neurons used in the hidden layer as it allows connections between all layers [Hunter and Wilamowski, 2011, Wilamowski et al., 2008].

The HMLP architecture consists of input, hidden and output layer with the same functions as the one in standard MLP architecture. Each node in input layers is connected to each node in hidden and output layer, and each node in hidden layer is connected to output layer forming a fully connected network. Since HMLP is a variant of MLP architecture, the network architecture can be constructed using multiple hidden layers; however there are usually no significant advantages or improvements on model prediction performance [Tu, 1996, Samal, 2009]. For a single hidden layer case, the outputs formulation from hidden and output layer of an HMLP network is given as follows:

$$v_h(t) = g_h\left(\sum_{j=1}^{m} W1_{hj} X_j(t) + B1_h\right) \qquad \text{for } h = 1, 2, 3 \cdots H \quad (4.6)$$

$$\hat{y}_i(t) = g_i\left(\sum_{h=1}^{H} W2_{ih}V_h(t) + B2_i\right) + g_i\left(\sum_{j=1}^{m} W3_{ij}X_j(t)\right) \quad \text{for } i = 1, 2, 3 \cdots n \quad (4.7)$$

where m, n and H denote the number of inputs, outputs and hidden nodes of the HMLP network respectively, $B1_h$ and $B2_i$ is the bias elements of input and output layer and $X_j(t)$ denotes the inputs data fed into the HMLP network. The weight matrix that connects the input layer to the hidden layer is given by $W1_{hj}$ and $W2_{ih}$ indicates weight matrix that connect the hidden layer to the output layer. The linear connections of HMLP that connect input and output layer are represented by $W3_{ij}$. The function g_h and g_i are the activation function used in the hidden and output layer similar to MLP architecture. In our work, the hyperbolic tangent and linear activation function are used in the hidden and output layer respectively. The reason to choose hyperbolic tangent as activation function in neural network model training is because of its superior learning speed and accuracy compared with other activation functions such as Bipolar Sigmoid, Unipolar Sigmoid, Conic Section and Radial Basis Function [Karlik and Olgac, 2010].

Both MLP and HMLP networks can be used for prediction or forecasting time series by presenting inputs to the network with lags of variables to be predicted. This would create a set of input variables with delays being fed to the network input layer, which in turn serve as a short term memory built into network's weights. This modelling method using feed-forward network with time lagged inputs is also known as Neural Network based Autoregressive structure with extra inputs (NNARX) model and its formulation and representation are given in Section 4.3.2.

4.2.3 Elman Network

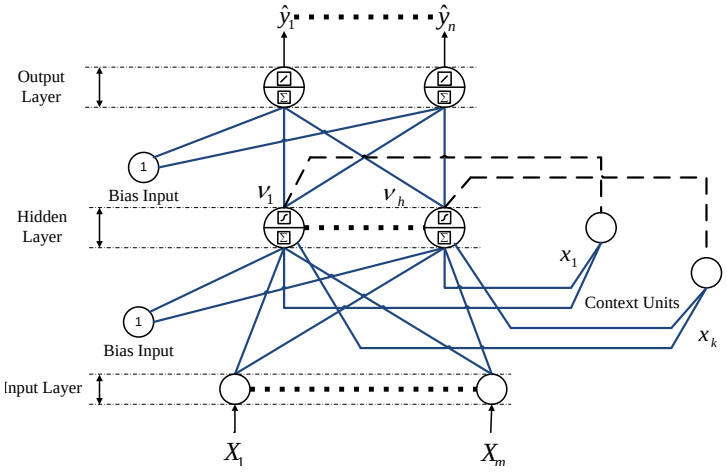
The Elman network is a simple dynamically driven recurrent network which attempts to capture long term history in measurement data [Samarasinghe, 2007]. Elman network was first introduced by Elman [1990] and it consists of a feed-forward network where hidden neuron signals are copied to a context/memory units and fed back into the network in the next time step. The Elman network realised the long term history of data through its internal memory without relying on externally provided memory as in NNARX network. Using this recurrence loop concept, the network can develop

memories of the past activation of the hidden units. The Elman network offers benefits over NNARX network in such a way that the regression structure and memory dynamics are determined by the network itself [Samarasinghe, 2007]. Thus, it is not necessary to provide the time lagged inputs to the network as in NNARX network.

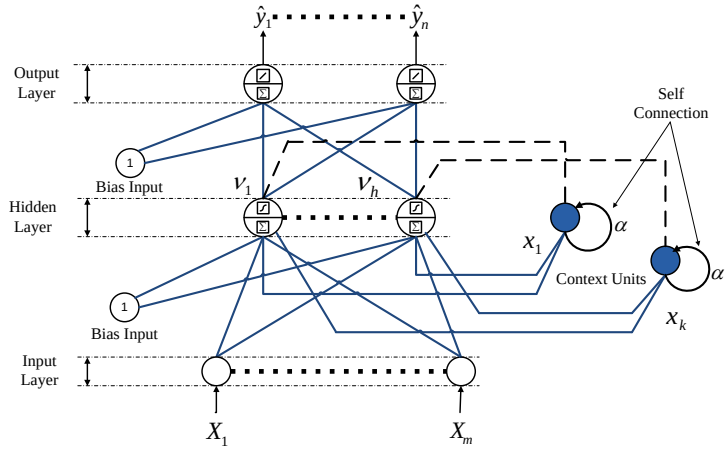
Figure 4.6(a) shows the basic Elman network which has similarity to typical feed-forward MLP network in term of input, hidden and output layers. The external inputs, X_j from the measurements are fed to the input layer and propagated forward to the hidden neurons. The outputs of from the hidden neurons, v_h are then sent forward to the output layers to produce the network output predictions, y_i at the next time step $t+1$. The Elman network also consists of extra processing units called context units. The context units receive and store the output signals from the hidden neurons and with a one step delay [Pham and Liu, 1993]. At the next time step, the outputs of context units, x_k would send the delayed hidden neuron outputs to the hidden neurons. Note that solid lines indicate weights that are allows to adapt according to an optimisation routine, while dashed lines denotes recurrent connections from hidden neurons to context units which are not modifiable and usually their strengths are fixed as 1.

Findings from Pham and Liu [1993] have suggested that the basic Elman network was found unable to identify higher order linear or non-linear dynamic system due to insufficient memory in the network. Pham and Liu [1996] proposed a modified version of Elman network to increase the memory capabilities of the basic Elman network though the use of self-connections in the context units. Figure 4.6(b) shows that the modified version of basic Elman network with self-connections in the context units indicate by scalar value α. The gradient calculation of the modified Elman network has similarity in structure to gradient calculation from dynamic back-propagation (DBP) or back-propagation through time (BPTT) algorithm [Pham and Liu, 1996]. This algorithm provides a dynamic trace of gradients in parameter space and would enable the network to model dynamics system of higher order.

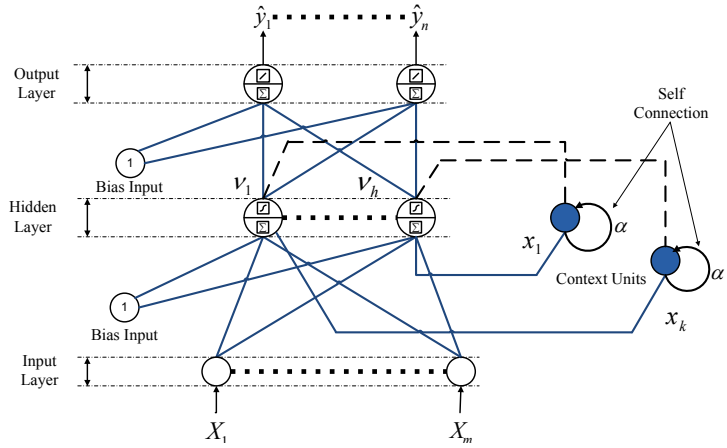
In this work, the modified version of the basic Elman network is used for system identification of helicopter dynamics system. Further modification has been made to the modified Elman network proposed in Pham and Liu [1993] by reducing the number of



(a)



(b)



(c)

Figure 4.6 The structure and operation of Elman network. (a) The basic Elman Network (b) The modified version of Elman network with self-connections in the context units; and (c) The modified version of Elman network with reduced weight connections.

connections from context units to hidden units. Figure 4.6(c) shows the architecture of the modified Elman network with reduced weight connections. Note that the connections from context units to hidden units are made with one to one connections instead of multiple connections from a single context unit to multiple hidden units as in Figure 4.6(b). For a single hidden layer case, the outputs formulation from hidden, context unit and output layer of a modified Elman network are given as follows:

$$v_h(t) = g_h\left(\sum_{j=1}^{m} W1_{hj} X_j(t) + B1_h + \sum_{k=1}^{h} W3_k x_k(t)\right) \quad \text{for } h = 1, 2, 3 \cdots H \tag{4.8}$$

$$x_k(t) = v_h(t-1) + \alpha x_k(t-1) \quad \text{for } k = 1, 2, 3 \cdots H \tag{4.9}$$

$$\hat{y}_i(t) = g_i\left(\sum_{h=1}^{H} W2_{ih} v_h(t) + B2_i\right) \quad \text{for } i = 1, 2, 3 \cdots n \tag{4.10}$$

Similar to MLP and HMLP network architecture, variables m, n and H denote the number of inputs, outputs and hidden nodes of the Elman network respectively, $B1_h$ and $B2_i$ is the bias elements of input and output layer and $X_j(t)$ denotes the inputs data fed into the Elman network. The function g_h and g_i are the activation functions used in the hidden and output layer similar to MLP architecture. The weight connections that connects the input layer to the hidden layer is given by matrix $W1_{hj}$. Matrix $W2_{ih}$ indicates weight matrix that connect the hidden layer to the output layer. The weight connections from context units to hidden neurons are represented by vector $W3_k$. The self-connections α in the context units are set to be the same for all context units and typically selected between $0 \leq \alpha \leq 1$. The higher the value of α indicates that network has the higher capability to trace the gradient further into the past [Pham and Liu, 1993].

4.3 SYSTEM IDENTIFICATION WITH NEURAL NETWORK

In this section, the neural network based system identification procedures and the description of learning algorithms used for NN trainings are presented. The overall neural network modelling approach used in this work consists of several steps including the test data gathering process, neural network model structure selection, model estimation and lastly the validation process as shown in Figure 4.7. The purpose of the test data gathering process is to collect a set of data from the system behaviour over the entire operating conditions which later will be used in processes to infer a dynamic model of the system. The type of excitation signal, data collection, sampling frequency and filtering option will be discussed further in Section 4.3.1. In the model structure selection process, the users need to decide the model structure to be used before the estimating process. In this work, we use the MLP network architecture based on ARX (Autoregressive structure with extra inputs) model structure for its simplicity and stable prediction [Norgaard, 2000]. For the model structure selection process, issues of selecting the proper lag space size and the minimum number of neurons to satisfy the average generalisation error is given in Section 4.3.2. The minimisation of error criterion is done using the Levenberg-Marquardt algorithm with added regularisation term to prevent an over-fitting problem. For the final process, the model estimate is validated against one-step ahead predictions, k-step ahead predictions, correlation between prediction errors and measured outputs and reliability visualisation of the predictions.

4.3.1 Collection of Flight Test Data

The flight test was conducted on the UAV helicopter platform in calm weather conditions. Different flight manoeuvres were conducted to excite the desired dynamic of interest. For example, after the helicopter reached steady and level condition, the yaw dynamics was excited using only tail collective pitch command while other input commands were used to balance the helicopter in such a way to make the vehicle oscillate roughly around the operating point of interest.

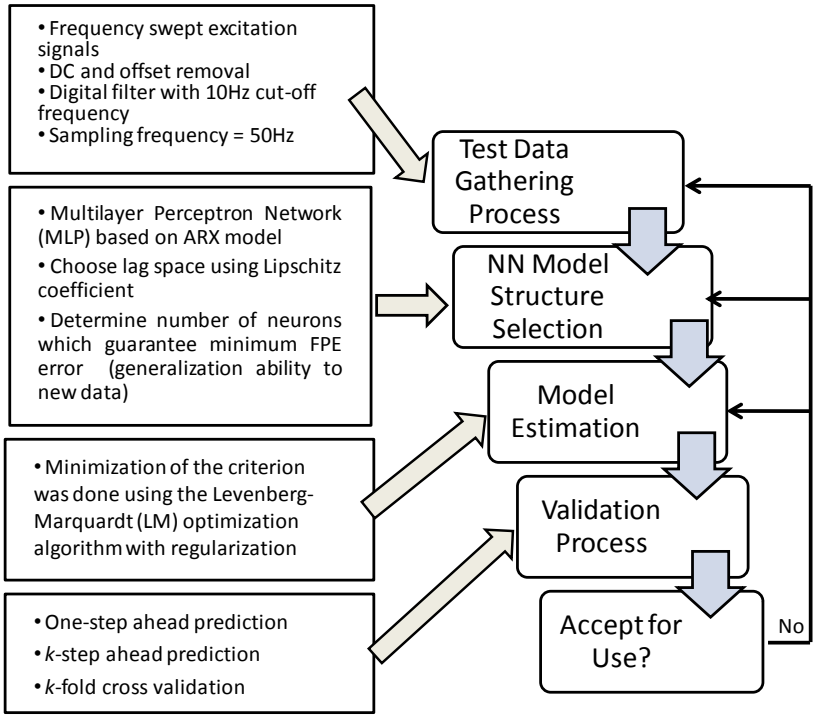


Figure 4.7 Overview of neural network based system identification procedure.

Training and validation data were collected from specifically designed frequency swept excitation signal suggested in Tischler and Remple [2006], Wang et al. [2011]. This type of signal is commonly used to collect experimental flight data in aircraft and rotorcraft system identification. The frequency swept excitation signal is not required to have constant amplitude. It is recommended that the pilot executes two good low frequency cycle inputs (20 s) and then gradually increase the swept frequency to mid and higher frequencies before ending the manoeuvre in the trim position. Starting and ending the record in aircraft trim state enables concatenating flight data collected from several test runs while at the same time ensuring rich signal content.

All measurements of the helicopter's state variables were collected using an inertial measurement unit (IMU) where the data that were recorded during the test were Euler angles: roll ϕ, pitch θ and yaw Ψ; angular rates in body coordinate frame: roll rate, p, pitch rate, q and yaw rate, r and body accelerations: A_x, A_y, A_z. The control inputs measured during the experiment were the stick deflection from the pilot's collective pitch δ_{col}, tail pedal δ_{ped}, longitudinal cyclic δ_{lon} and lateral cyclic δ_{lat}. The four servomotor

signals $s = [s_{ail} \quad s_{aux} \quad s_{ele} \quad s_{rud}]^T$, can be translated to pilot's stick positions (Input δ range $= \pm 1$), $\delta = [\delta_{lon} \quad \delta_{lat} \quad \delta_{col} \quad \delta_{ped}]^T$, by means of a linear transformation:

$$\delta = G^{-1}(s - s_{trim}) \tag{4.11}$$

where s_{trim} are the servo signals at trim values which indicate the necessary pulse width values to level the swash plate position and tail pitch cyclic. Matrix G (mixing gains) has to be determined through the measurement of servo signals for different stick positions to get the exact relationship between pulse width commands sent to the servos and the requested control inputs. During the system identification experiments, a separate micro-controller was used to continuously measure the servo signals. It passed the values through to the MBS270 over a Universal Asynchronous Receiver/Transmitter (UART) serial interface. The overall system architecture used in system identification experiment can be referred in Section 3.5. The resulting values for matrix G and s_{trim} was found to be:

$$G = \begin{bmatrix} 0.049 & 0.126 & -0.239 & 0 \\ -0.047 & 0.127 & 0.248 & 0 \\ 0.103 & 0.003 & 0.242 & 0 \\ 0 & 0 & 0 & -0.256 \end{bmatrix}; \quad s_{trim} = \begin{bmatrix} 1.387 \\ 1.692 \\ 1.387 \\ 1.503 \end{bmatrix} \tag{4.12}$$

The common frequency range for the excitation signal used in rotorcraft system identification and control is between 0.3 rad/s to 20 rad/s. It is also recommended in Tischler and Remple [2006] that an identical filter to be used for all output and input signals with a cut-off frequency 5 times higher than the maximum excitation signal frequency. Hence to reduce the noise in sensors data, the cut-off frequency of the low pass filter used in this study was selected at 15 Hz. The sampling rate of the sensors was selected at 100 Hz which was at least 25 times higher than the maximum excitation frequency.

The IMU (3DM-GX1) filters the raw sensor outputs on-board, combining the data from the accelerometers, gyroscopes, and magnetometers. However since the position

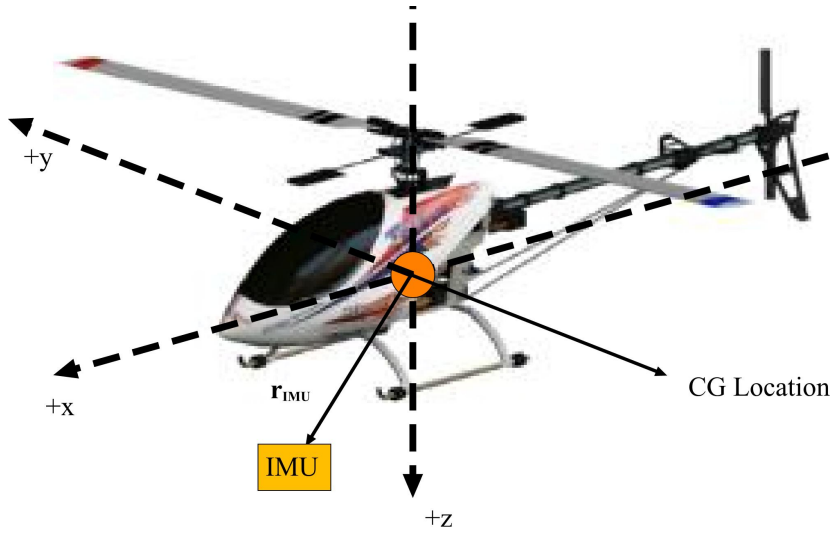


Figure 4.8 Offset distance of the IMU with respect to centre of gravity.

of the IMU was placed not exactly at the centre of gravity (CG) of the vehicle, the accelerometer only senses the acceleration at the attachment point of the avionics housing. Therefore, the linear accelerations and velocities have to be corrected for the offset between the IMU position and the CG (Figure 4.8) before being used for system identification. Tischler and Remple [2006] derived the correction of acceleration measurement at the centre of gravity of the vehicle using kinematics of relative motion as follows:

$$A_{x_{cg}} = A_x - \dot{q}z_a\dot{r}y_a - q\left(py_a - qx_a\right) + r\left(rx_a - pz_a\right) \tag{4.13}$$

$$A_{y_{cg}} = A_y - \dot{p}z_a + \dot{r}x_a - p\left(py_a - qx_a\right) + r\left(qz_a - ry_a\right) \tag{4.14}$$

$$A_{z_{cg}} = A_z - \dot{p}y_a + \dot{q}x_a - p\left(rx_a - pz_a\right) + q\left(qz_a - ry_a\right) \tag{4.15}$$

When the location of the vehicle's CG (x_a, y_a, z_a) was known, the acceleration measurement from the accelerometer was corrected for the effect of sensor offset. A simple weight balancing test was carried out to determine the location of the CG which gave the measured offset values as $r = [x_a \quad y_a \quad z_a]^T = [0.043 \quad 0 \quad 0.153]^T$ m. As mentioned in Mettler [2003], the effect of acceleration biases due to sensor offset will introduce a large bias reading in the y-body direction when a lateral cyclic sweep is applied.

4.3.2 Neural Network Model Structure Selection

The primary objective of system identification process is to create a model describing the underlying relationships between input-output variables. There are two types of NN model structure that have been successfully used for system identification and time-forecasting tasks: a) NNARX model structure or also known as time lagged feed-forward networks; b) dynamically driven recurrent networks.

In a neural network based ARX (NNARX) model structure, the variable to be estimated and other influencing variables including their time lags are typically fed into a static feed-forward network such as multi-layer perceptron (MLP) network [Norgaard, 2000]. The reason to include extra variables from the time lags is to enable the model to extract information from the time lags that are not included in the current measurement and the lags of the variables of interest [Samarasinghe, 2007]. This would subsequently improve the prediction accuracy. In contrast to NNARX model structure, a dynamically driven recurrent network such a Elman network learns the dynamics of a time series through internal feedback connections that remember the past state of the series.

The black box based modelling approach such as NNARX is usually used to deduce the dynamic model of a system by taking into account the relationship between all inputs and outputs of the system. Generally, we represent the n^{th} order discrete time helicopter non-linear with m inputs, p outputs as follows:

$$\begin{aligned} x\left(t+1\right) &= h\left[x(t), u(t)\right] \\ y(t) &= g\left[x(t)\right] \end{aligned} \tag{4.16}$$

where $x \in \mathbb{R}^p$ is the state vector, $y \in \mathbb{R}^n$ is output vector and $u \in \mathbb{R}^m$ is input vector at discrete time step t with assumption that the dynamic system has m inputs, p states and n outputs.

The model structure for neural network modelling used in this research was adapted from standard ARX (Autoregressive structure with extra inputs) model structure reported in Ljung [1999]. Using this approach, the measurement data are sampled in discrete time step, t and the sampled values between input and output measurements

are related through a linear differential equation. Considering single input and single output (SISO) case, the input-output relationship of a linear dynamic system described in this research is given as follows:

$$y(t) + a_1 y(t-1) + \cdots + a_{n_y} y(t-n_y) = b_1 u(t) + b_2 u(t-2) + \cdots + b_{n_u} u(t-n_u) + v(t) \tag{4.17}$$

where n_y and n_u are the sizes of past output and input observations and $v(t)$ is the system disturbance. Then the general polynomial equation (4.17) can be rewritten in terms of the time shift operator q^{-1} as:

$$A(q^{-1})y(t) = B(q^{-1})u(t) + v(t) \tag{4.18}$$

where $A(q^{-1})$ and $B(q^{-1})$ are polynomials in the time shift operator:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{n_y} q^{-n_y} \\ B(q^{-1}) &= 1 + b_1 q^{-1} + \cdots + b_{n_u} q^{-n_u} \end{aligned} \tag{4.19}$$

The model in (4.17) or (4.18) describes the dynamic relationship between the input and the output signals. Similar to definition in Section 4.2.1, the input-output relationship (coefficient of ARX model) and the lagged input output data is express in terms of parameter vector θ and time regression vector $\varphi(t)$ as:

$$\theta = \begin{bmatrix} a_1 & a_2 & \cdots & a_{n_y} & b_1 & b_2 & \cdots & b_{n_u} \end{bmatrix}^T \tag{4.20}$$

$$\varphi(t) = \begin{bmatrix} y(t-1) & \cdots & y(t-n_y) & u(t-1) & \cdots & u(t-n_u) \end{bmatrix}^T \tag{4.21}$$

The equation (4.17) then can be rewritten for k-step ahead prediction as:

$$\hat{y}\,(t+k\,|t,\theta\,) = \varphi^T(t+k)\theta + v(t) \tag{4.22}$$

This model describe the observed output variable $y(t)$ as an unknown linear combination of the component of the observed time regression vector φ with system disturbance

$v(t)$ [Ljung, 1999]. It is a well known type of system model and also known as linear regression in statistics.

Since we are considering a system that is non-linear in nature, the non-linear term $h(.)$ can be introduced to the linear ARX model predictor and the resulting non-linear ARX model structure with k-step predictor is denoted as:

$$\hat{y}\,(t+k\,|t,\theta\,) = h\left[\varphi^T(t+k)\theta\right] \tag{4.23}$$

Parameter vector θ in Equation (4.23) contains adjustable parameters which can also be represented in the neural network model as weight connections. The processing neurons in the hidden layer allow the NN model to learn the non-linear relationship between the measured outputs and inputs. The NN representation of the non-linear ARX model structure is known as Neural Network ARX (NNARX) and it is also known as Serial-Parallel model or focus time lagged feed-forward network in NN community [Samal, 2009, Samarasinghe, 2007].

The fully connected MLP or HMLP architecture containing only one hidden layer discussed in Section 4.2.1 and 4.2.2 can be chosen to learn the non-linear relationship of the NNARX model. The conceptual diagram of NNARX model structure using MLP or HMLP network are given in Figure 4.9(a) and 4.9(b). The output calculation from the MLP structure in Equation (4.1) and (4.2) is reproduced here to represent the NNARX predictor formulation for a single hidden layer MLP network:

$$\hat{y}\,(t\,|\theta\,) = h_i\,(\varphi,\theta) = F_i\left(\sum_{h=1}^{H} W_{ih} f_h\left(\sum_{j=1}^{m} w_{hj}\varphi_j(t) + b_h\right) + B_i\right)$$
$$\text{with} \quad h = 1,2,3\cdots H \qquad \text{and} \qquad i = 1,2,3\cdots n \tag{4.24}$$

and the parameter and regression vectors are given by:

$$\theta = \begin{bmatrix} w_{hj} & W_{ih} & b_h & B_n \end{bmatrix} \tag{4.25}$$

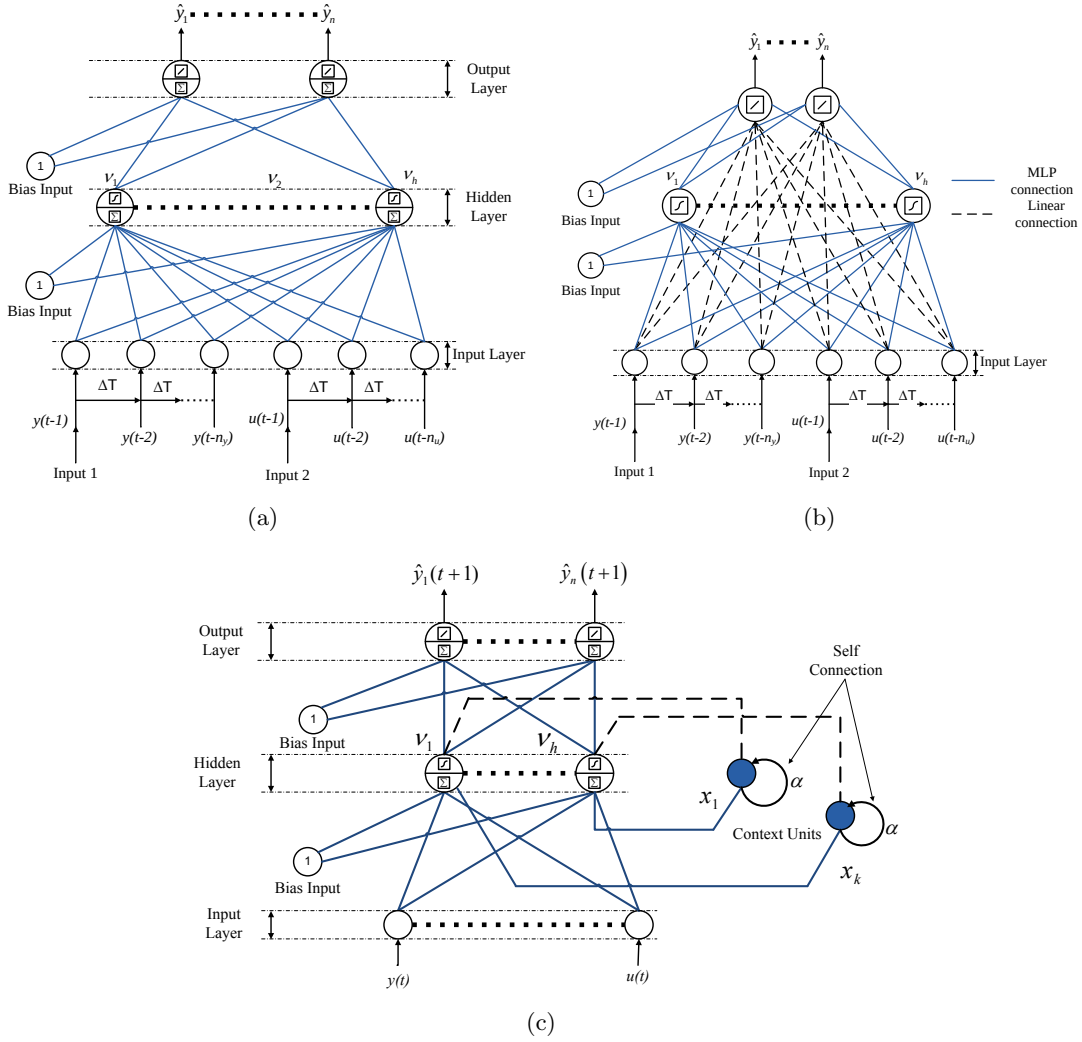


Figure 4.9 The model structure using different ANN networks architecture (a) The NNARX based model structure using MLP network (b) The NNARX based model structure using HMLP network; and (c) Prediction/Forecasting using modified Elman network with reduced weight connection from context units to hidden units.

$$\begin{aligned}\varphi(t) &= \begin{bmatrix}\varphi_1 & \varphi_2 & \cdots & \varphi_m\end{bmatrix} \\ &= \Big[y(t-1) \quad y(t-2) \cdots y(t-n_y) \\ &\qquad u(t-1) \quad u(t-2) \cdots u(t-n_u)\Big]\end{aligned} \tag{4.26}$$

where w_{hj} is the weights matrix between the input layer and the hidden layer and W_{ih} is the weights matrix between the hidden layer and the output layer. The functions $f_j(*)$ and $F_i(*)$ are non-linear activation function for neurons in each hidden and output layers. The symbol H denotes the number of neurons in the hidden layer while $b1$ and $b2$ are the bias elements for the input layer and output layer. The number of inputs

and outputs of the neural network are presented by m and n respectively. Similarly, the NNARX model can also be represented using HMLP network by introducing the lagged time input variables into the neural network model.

The Elman network with modification in the network's internal dynamics is found capable of identifying an n^{th} order discrete dynamic system in Equation (4.23), through empirical findings in Pham and Liu [1993]. Furthermore, Pham and Liu [1996] suggested that in order to model the dynamic system in Equation (4.23) from experimental data using MLP network, a $n_y + n_u$ input (regressor) nodes are needed to be included in the network. However, if an Elman network is used to model a NNARX model, only current measurement data are needed to be fed into the input nodes as shown in Figure 4.9(c). Therefore, the modified Elman network is significantly smaller in size compared with MLP or HMLP network when large time lags (model order) are used.

4.3.2.1 Lag Space Selection for Feed-Forward MLP or HMLP Network

After deciding the input parameters to be used in the model, the number of past inputs and outputs fed into the MLP or HMLP neural networks were decided based on the calculation of the Lipschitz coefficient given in Norgaard [2000]. Using this coefficient, it is possible to determine the proper lag space via experimental data. The sizes of output and input time regression vectors depend on the degree of non-linearity of the Lipschitz coefficients where an insufficient number of regressors will result in high Lipschitz coefficients and small numerical values for extra regressors.

The Lipschitz coefficient is calculated using the following formula for each input $u_i(t)$ and output $y_j(t)$ pairs:

$$q_{ij} = \left| \frac{y(t_i) - y(t_j)}{\varphi(t_i) - \varphi(t_j)} \right| \qquad i \neq j \tag{4.27}$$

The outline of Lipschitz coefficient approach is given as follows:

1. Determine the Lipschitz quotients for all combination of input-output pairs using (4.27) for a given choice of number of past outputs and inputs.

2. Select the p largest quotients with p selected as $p = 0.01N \sim 0.02N$.

3. Calculate criterion according to:

$$\bar{q}_{ij}^{(n)} = \left(\prod_{k=1}^{p} \sqrt{n} q_{ij}^{(n)}(k) \right)^{\frac{1}{p}}$$

   where $n = n_y + n_u$

4. Repeat step (1)-(3) for different lag structures.

5. Plot the calculated criterion as a function of number of past outputs and past inputs (lag space).

4.3.3 Off-line and Recursive Methods

The system identification for inferring the helicopter dynamic model can be conducted using off-line (batch) and recursive based system identification methods. The estimation of a dynamic model in off-line neural network identification method involves the training process being carried out over some finite data gathered beforehand. Over the whole length of the data record, we determine the best weights (parameters vector θ) that give the best fit for the measurement data over repetitive iterations. Obviously, the off-line methods have a disadvantage such as their unsuitability for tracking time varying dynamics, as the amount of computation time for the training phase in each iteration might exceed the available processing time [Norgaard, 2000]. The adaptive control is an example where a model needs to be identified at the same time as a control law is calculated to compensate the time varying control variables. Even though the off-line methods are not suitable for real-time implementation, several researchers have used a mini-batch data size for off-line method implementation in real-time as proposed in Samal [2009], Puttige and Anavatti [2006], Puttige [2009]. However, in these examples, the NN estimation and control were restricted only for SISO case and smaller network due to limited computation capabilities to invert big Hessian matrix at each iteration.

To overcome the disadvantages of off-line methods, the recursive based system identification methods can be used in tracking time varying dynamics. The recursive

model estimation is a system identification technique that enables us to infer a model that adapts to time-varying dynamics based on real-time data coming from the system. In contrast to batch training methods, the recursive methods enforce update to NN parameter vector based only on a single data set at the current sample t. To achieve real-time implementation of neural network based system identification, the estimation of neural network's parameter vector θ can be carried out using recursive algorithms as described in Billings et al. [1992], Norgaard [2000], Youmin and Li [1999], Ljung and Soderstrom [1983], Asirvadam [2008], Shamsudin and Chen [2012b]. Norgaard [2000] has suggested that the recursive identification algorithms have several advantages over the batch methods. The implementation of the method is simpler, less memory-consuming with faster convergence since the redundancy in data set is effectively utilised.

Recursive algorithm can also be implemented similar to the off-line training, where the recursive training is repeated several time on the finite training set Z_N collected in advance [Norgaard, 2000, Billings et al., 1991, 1992]. Figure 4.10 shows the difference between the batch algorithm, mini-batch algorithm, on-line recursive algorithm and repeated recursive algorithm. The parameter vector θ updating process usually starts with initial random weights and it is carried out forward to the next iteration as computation progresses. The implementation of batch and mini-batch algorithm is similar but differs in the number of data samples used for training. In the recursive algorithm methods, the parameter vector update is obtained in real-time as the measurement data become available from the instrumentation system. Figure 4.10(d) shows the implementation of recursive algorithm as an off-line method. The parameter vector θ is updated at each time sample t over a fixed length data sample. At the end of first iteration, the last parameter vector θ is used as the initial update to the second iteration step. This iteration process will stopped if the mean square criterion converges to pre-defined threshold as in batch training algorithm implementation.

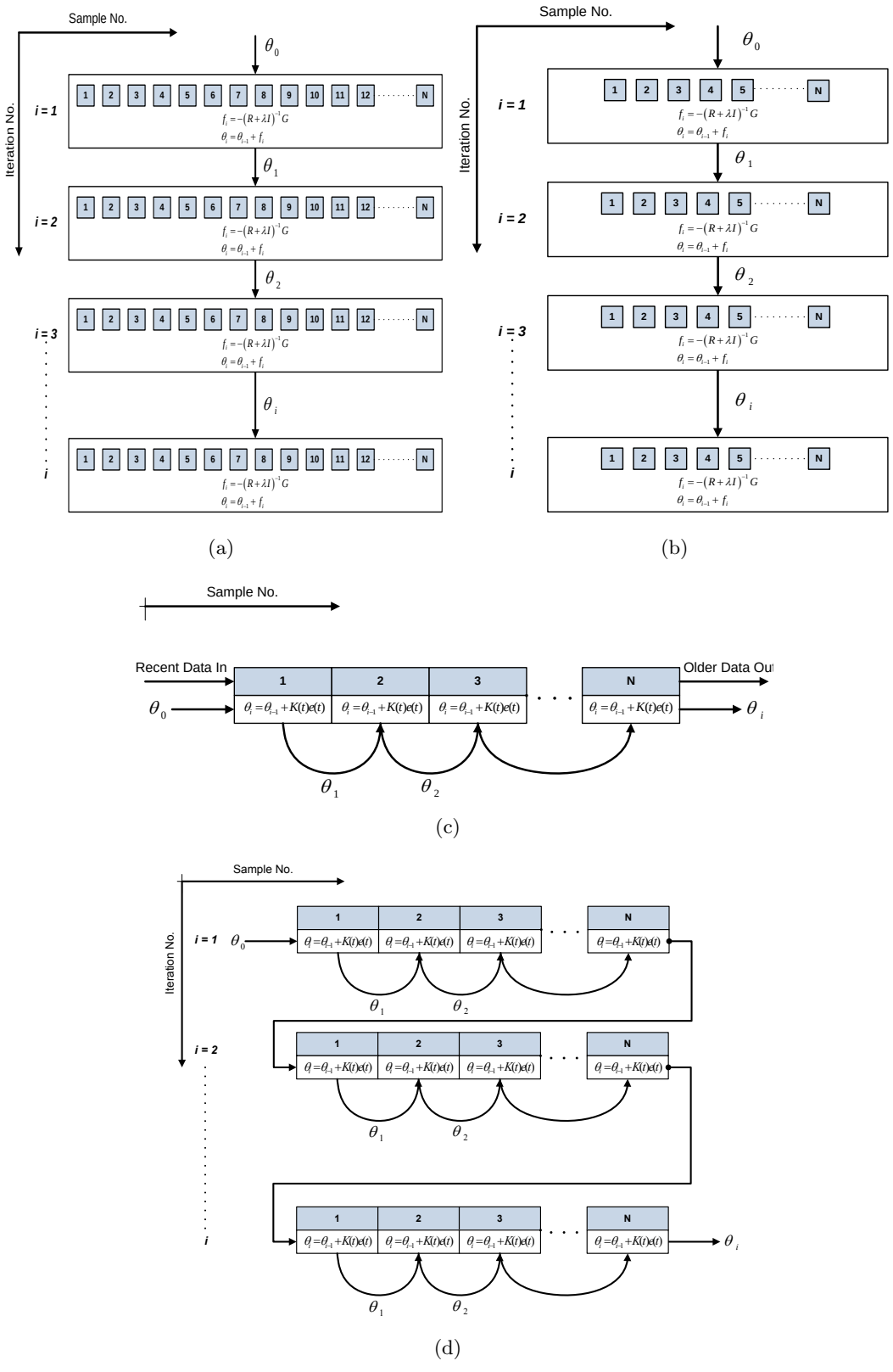


Figure 4.10 The different types of Neural Network model estimation methods: (a) Batch Algorithm; (b) Mini-batch Algorithm; (c) Recursive Algorithm; and (d) Repeated Recursive Algorithm

4.3.4 Off-line based Neural Network Model Estimation

After selecting the model structure, the next step in the system identification process is to determine the best weights (vector parameters θ) that give the best fit between the NNARX model and measurement data. This is achieved by minimisation of error cost function. As mentioned in Norgaard [2000], the measure of prediction's closeness to the true outputs of the system is given by mean square error type criterion as:

$$V_N\left(\theta, Z_N\right) = \frac{1}{2N}\sum_{t=1}^{N}\left[y\left(t\right) - \hat{y}\left(t\,|\theta\right)\right]^2 = \frac{1}{2N}\sum_{t=1}^{N} e^2(t,\theta) \tag{4.28}$$

with linear approximation of prediction error given by:

$$e\left(t,\theta\right) = y\left(t\right) - \hat{y}\left(t\,|\theta\right) \tag{4.29}$$

where N is the number of input-output pairs used for training, $y(t)$ is the real measurement output of the system, $\hat{y}(t|\theta)$ is the predicted output vector and the training data set is given by $Z_N = [y(t), u(t)]$. For multiple input-output case (n outputs), the measurement output of the system $y(t)$ and predicted output $\hat{y}(t|\theta)$ will became a $n \times N$ matrix which would produced a vector of mean square error criterion.

The solution involving the quadratic criterion in Equation (4.28) is also known as *ordinary non-linear least squares* problem which is a part of unconstrained optimisation study [Norgaard, 2000, Sofer et al., 2009]. The minimisation of criterion is carried out using numerical search procedure starting out from initial guess of parameter vector $\theta^{(0)}$. The weights of the network are then adjusted according to some training methods and stopped after error criterion evaluation achieves certain threshold. Typically, most off-line minimisation iterative schemes have the following general form:

$$\theta^{(i+1)} = \theta^{(i)} - \mu^{(i)}\left[H^{(i)}\right]^{-1} V_N'\left(\theta^{(i)}, Z_N\right) \tag{4.30}$$

where $\theta^{(i)}$ is the parameter vector estimation at i^{th} iteration, $H^{(i)}$ is the matrix that modifies the local search direction defined by $V_N'\left(\theta^{(i)}, Z_N\right)$ and constant $\mu^{(i)}$ denotes the step size to assure that $V_N\left(\theta^{(i)}, Z_N\right)$ decreases from previous update.

In order to minimise the cost function in Equation (4.28), the Levenberg-Marquardt (LM) iterative search algorithm was used for the neural network training process. The LM algorithm is considered the best choice of training method in many NN applications particularly for training small to medium size networks [Naghdinezhad et al., 2006, Wilamowski and Hao, 2010, Wilamowski, 2009, Garratt and Anavatti, 2012]. The problems of criterion minimisation in NN are often ill-conditioned as reviewed in Saarinen et al. [1993] and Ngia and Sjoberg [2000]. As a remedy to the problem, the LM algorithm adds positive diagonal matrix λI to improve the numerical conditioning to the Hessian matrix [Norgaard, 2000, Ngia and Sjoberg, 2000]. Typically, the LM optimisation process is carried out iteratively over a given data set to achieve the minimum error criterion. The LM optimisation algorithm uses the Gauss-Newton's Gradient vector $G(\theta)$ and Hessian matrix $R(\theta)$, which were derived specifically for the neural network model in Yu and Wilamowski [2011], Norgaard [2000]. These important matrices are represented in the following equations:

$$G\left(\theta\right) = V_N^{'}\left(\theta, Z_N\right) = \frac{1}{N}\sum_{t=1}^{N}\psi\left(t|\theta\right)\left[y\left(t\right) - \hat{y}\left(t|\theta\right)\right] \tag{4.31}$$

$$R\left(\theta\right) = V_N^{''}\left(\theta, Z_N\right) = \frac{1}{N}\sum_{t=1}^{N}\psi\left(t|\theta\right)\left[\psi\left(t|\theta\right)\right]^T \tag{4.32}$$

where $\psi(t|\theta)$ is the matrix that represent the first order partial derivative of the one-step ahead prediction with respect to the parameter vector θ and it is also known as the Jacobian matrix. The complete formulations of Jacobian matrix for MLP, HMLP and Elman networks are given in Section 4.3.4.1. The Hessian matrix $R(\theta)$ has a dimension of $d \times d$ and Gradient vector $G(\theta)$ with dimension of $d \times 1$, where d is the total number of elements (weights + biases) in the parameter vector θ.

For the LM iterative algorithm, the step size $\mu^{(i)}$ in Equation (4.30) is set to $\mu^{(i)} = 1$ and the $H^{(i)}$ matrix is replaced with the following equation:

$$H^{(i)} = R^{(i)}(\theta) + \lambda^{(i)} I \tag{4.33}$$

where $R^{(i)}(\theta)$ is defined as $V_N^{''}\left(\theta, Z_N\right)$ in Equation (4.32). Finally, we could find the

minimum of the error criterion by iteratively solving the following update equation:

$$\theta^{(i+1)} = \theta^{(i)} - \left[R^{(i)}(\theta) + \lambda^{(i)} I\right]^{-1} G^{(i)}(\theta) \tag{4.34}$$

where I is the identity matrix with a size equal to Hessian $R(\theta)$ matrix and the $\lambda^{(i)}$ constant is a damping factor used for deciding the step size. The value of $\lambda^{(i)}$ is selected to be $\lambda^{(i)} \geq 0$. The usage of the constant $\lambda^{(i)}$ can also be viewed as a blending factor between Gauss-Newton (GN) and Steepest Descent (SD) update, which provides LM algorithm with fast convergence speed of the GN algorithm and the stability of SD method [Yu and Wilamowski, 2011]. The largest update can be possibly achieved by choosing $\lambda^{(i)} = 0$ which gives a full GN step. Shorter step length as in SD algorithm is achieved by taking $\lambda^{(i)} \to \infty$ which will cause the diagonal elements of $R^{(i)}(\theta)$ to dominate [Ngia and Sjoberg, 2000].

In order to determine λ, the indirect method used in Norgaard [2000] and Fletcher [1981] is adopted by calculating the following ratio to determine the accuracy of approximation:

$$r^{(i)} = \frac{2\left[V_N\left(\theta^{(i)}, Z_N\right) - V_N\left(\theta^{(i)} + f^{(i)}, Z_N\right)\right]}{\underbrace{\left(f^{(i)}\right)^T G\left(\theta^{(i)}\right) + \left(f^{(i)}\right)^T f^{(i)} \lambda^{(i)}}_{\text{reduction approximation}}} \tag{4.35}$$

The main purpose of introducing the ratio calculation is to measure how well the reduction of the criterion $V_N(\theta, Z_N)$ matches the reduction predicted by approximation terms in denominator of ratio calculation in Equation (4.35). The damping factor λ is adjusted accordingly to the ratio $r^{(i)}$ by some factor [Norgaard, 2000]. The procedure for the LM algorithm using indirect method to determine λ is given in Figure 4.11. The reduction approximation (denominator term in Equation (4.35)) is most likely a close approximation to error criterion $V_N(\theta, Z_N)$, if the ratio $r^{(i)}$ value is close to one and parameter λ should be reduced by some factor. However, if the ratio $r^{(i)}$ is small or a negative value, parameter λ should be increased. Additional stopping criteria are normally introduced to this algorithm to prevent minimisation problems or to force early stopping such as stopping criteria based on maximum number of iterations, sum of square error that drops below a certain threshold, upper bound for gradient and

maximum weight change, maximum value of parameter λ or early stopping criterion due to training time constraint.

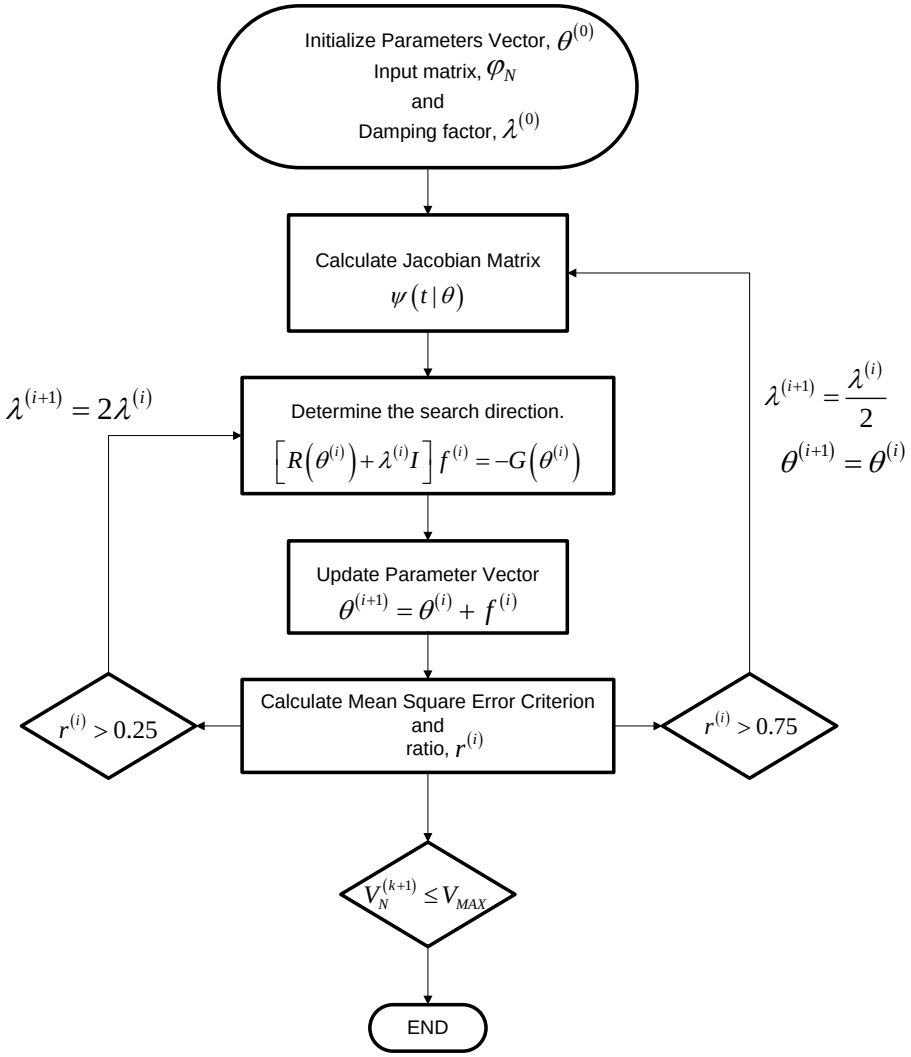


Figure 4.11 The Levenberg-Marquardt (LM) algorithm with step involving λ determination.

4.3.4.1 Jacobian Matrix Calculation

The calculation of Jacobian matrix $\psi(t|\theta)$ is an important step in Gradient or Newton based training algorithms. For the SISO case, the Jacobian matrix $\psi(t|\theta)$ is a $d \times N$ matrix where N denotes the number of samples in the training data set. The dimension

of the Jacobian matrix $\psi(t|\theta)$ would become $d \times (nN)$ if multiple input-output variables were considered in the estimation problem. The Jacobian matrix $\psi(t|\theta)$ also can be effectively calculated using an alternative approach such as finite differences method [Norgaard, 2000, Yu and Wilamowski, 2011, Wilamowski et al., 2008].

Consider a two layer MLP network with hyperbolic tangent hidden units and linear output units:

$$\begin{aligned}\hat{y}_i(t \mid \theta)_{mlp} &= \sum_{h=1}^{H}\left(W2_{ih}tanh\left(\sum_{j=1}^{m}W1_{hj}\varphi_j(t)+B1_h\right)+B2_i\right)\\ &= \sum_{h=1}^{H}W2_{ih}v_h(t)+B2_i\\ &\text{with}\quad h=1,2,3\cdots H\quad\text{and}\quad i=1,2,3\cdots n\end{aligned} \tag{4.36}$$

The MLP Jacobian matrix $\psi(t|\theta)$ is calculated by partial differentiating Equation (4.36) with respect to parameter vector θ arriving at the following results:

$$\psi\left(t|\theta\right)_{mlp}=\frac{\partial\hat{y}\left(t|\theta\right)}{\partial\theta}=\begin{cases}v_h(t) & \text{if } \theta=W2_{ih}\\ 1 & \text{if } \theta=B2_i\\ W2_{ih}\left(1-v_h^2(t)\right)\varphi_j(t) & \text{if } \theta=W1_{hj}\\ W2_{ih}\left(1-v_h^2(t)\right) & \text{if } \theta=B1_h\\ 0 & \text{otherwise}\end{cases} \tag{4.37}$$

The output prediction from the HMLP can also be expressed with the hyperbolic tangent hidden units and linear output unit as follows:

$$\begin{aligned}\hat{y}_i\left(t\,|\theta\right)_{hmlp} &= \sum_{h=1}^{H}\left(W2_{ih}tanh\left(\sum_{j=1}^{m}W1_{hj}\varphi_j(t)+B1_h\right)+B2_i\right)+\sum_{j=1}^{m}W3_{ij}\varphi_j(t)\\ &= \sum_{h=1}^{H}\left(W2_{ih}v_h(t)+B2_i\right)+\sum_{j=1}^{m}W3_{ij}\varphi_j(t)\\ &\text{with}\quad h=1,2,3\cdots H\quad\text{and}\quad i=1,2,3\cdots n\end{aligned} \tag{4.38}$$

where $W3_{ij}$ is the weights matrix of linear connection between input layer and output

layer. By performing differentiation as in MLP example, the Jacobian matrix for HMLP is given as:

$$\psi\left(t|\theta\right)_{hmlp} = \frac{\partial\hat{y}\left(t|\theta\right)}{\partial\theta} = \begin{cases} v_h(t) & \text{if } \theta = W2_{ih} \\ \varphi_j(t) & \text{if } \theta = W3_{ij} \\ 1 & \text{if } \theta = B2_i \\ W2_{ih}\left(1-v_h^2(t)\right)\varphi_j(t) & \text{if } \theta = W1_{hj} \\ W2_{ih}\left(1-v_h^2(t)\right) & \text{if } \theta = B1_h \\ 0 & \text{otherwise} \end{cases} \tag{4.39}$$

Similarly, the output prediction from the modified Elman network can also be expressed with the hyperbolic tangent hidden units and linear output unit as follows:

$$\begin{aligned} v_h(t) &= tanh\left(\sum_{j=1}^{m} W1_{hj}\varphi_j(t) + B1_h + \sum_{k=1}^{h} W3_k x_k(t)\right) \\ x_k(t) &= v_h(t-1) + \alpha x_k(t-1) \\ \hat{y}_i\left(t\,|\theta\right)_{Elman} &= \sum_{h=1}^{H} W2_{ih}v_h(t) + B2_i \\ &\text{with} \quad h = 1,2,3\cdots H \quad \text{and} \quad i = 1,2,3\cdots n \end{aligned} \tag{4.40}$$

where $W3_k$ is the weights vector for connections between context units and hidden layer. The vector $x_k(t)$ denotes the output of the context units. By performing the differentiation of network output with respect to each weight term, the Jacobian matrix for Elman network is given as follows:

$$\psi\left(t|\theta\right)_{Elman} = \frac{\partial\hat{y}\left(t|\theta\right)}{\partial\theta} = \begin{cases} v_h(t) & \text{if } \theta = W2_{ih} \\ W2_{ih}\left(1-v_h^2(t)\right)\varphi_j(t) & \text{if } \theta = W1_{hj} \\ W2_{ih}\frac{\partial v_h(t)}{\partial W3_k(t-1)} & \text{if } \theta = W3_k \\ 1 & \text{if } \theta = B2_i \\ W2_{ih}\left(1-v_h^2(t)\right) & \text{if } \theta = B1_h \\ 0 & \text{otherwise} \end{cases} \tag{4.41}$$

Pham and Liu [1996] has suggested that the internal feedback $x_k(t)$ has dependency on the previous weights $W3_k(t-1)$. This needs to be taken into account by updating the term $\partial v_h(t)/\partial W3_k(t-1)$ using the following formulation:

$$\frac{\partial v_h(t)}{\partial W3_k(t-1)} = \left(1 - v_h^2(t)\right) v_h(t-1) + \alpha \frac{\partial v_h(t-1)}{\partial W3_k(t-1)} \tag{4.42}$$

where α denotes the value of the self-connection in the context units. If the weight $W3_k$ changes are assumed to be small in each iteration, then the term $\partial v_h(t)/\partial W3_k(t-1)$ can be approximately written in recursive form as:

$$\frac{\partial v_h(t)}{\partial W3_k(t-1)} = \left(1 - v_h^2(t)\right) v_h(t-1) + \alpha \frac{\partial v_h(t-1)}{\partial W3_k(t-2)} \tag{4.43}$$

4.3.4.2 Training by Weight Regularisation

Implementation of NN model to predict or extract usable pattern from data requires special attention to several important aspects in model development such as examination of the generalisation ability, minimising model complexity, testing robustness of the model and selecting relevant inputs [Samarasinghe, 2007, Norgaard, 2000, May et al., 2011]. The generalisation performance of the trained NN model is analysed through the use of a validation data set which differs from the data set used for the training process. Generalisation indicates how well a trained model performs on a new data set. It is particularly important if a reliable prediction quality for new data is desired.

One of the main problems that may occur during neural network training is over-fitting [Sjoberg and Ljung, 1995]. This particular problem can be observed in network training where the error of the training data set could be reduced to a very small value, but when a new data set (validation data) is presented to the same model, the prediction error increase. The over-fitting problem usually happens due to the contribution of variance error which indicates an excessive number of neurons/weights in the network. On the other hand, if the model contains insufficient neurons or weights (under parametrised network), the bias error would dominate in such a situation. The situation in which model prediction under-fit (bias) and over-fit the validation data

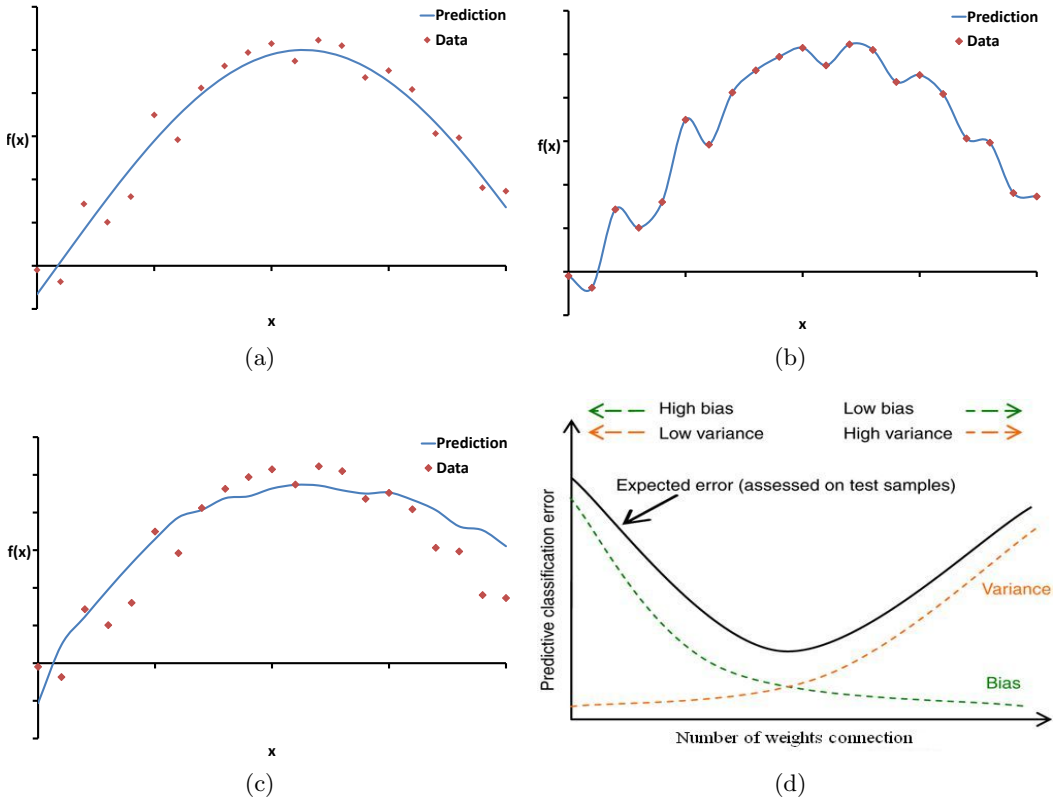


Figure 4.12 The bias-variance dilemma in prediction function: (a) Prediction from a model that generalise well on validation data; (b) Prediction from a model that over-fit the data due to excessive amount of free parameters (weights); (c) Prediction from a model that under-fit due to limited amount of free parameter (weights); and (d) A visualisation of the bias-variance error dilemma in model prediction. Figure adapted from Wang et al. [2008].

with noise is known as bias-variance dilemma which is shown in Figure 4.12. The effect of over-fitting, under-fitting and good generalisation performance of a prediction model is illustrated in Figure 4.12(a)–4.12(c). The overall description of effect of weights dimension on generalisation error is given in Figure 4.12(d). Thus, in order to obtain a reliable NN model with reduced generalisation error, a trade-off between these two extreme situations needs to be addressed.

To satisfy the bias-variance dilemma, the regularisation method has been proposed in this work to improve the generalisation ability of the neural network model. Regularisation method is an approach proposed to control the growth of weights during training to avoid over-fitting problems without relying on early stopping criterion [Sjoberg and Ljung, 1995, Samarasinghe, 2007, Norgaard, 2000]. The approach attempts to limit the flexibility of the network by introducing a regularisation term to augment the criterion

$V_N(\theta, Z_N)$ in Equation (4.28) such as:

$$W_N(\theta, Z_N) = \frac{1}{2N}\sum_{t=1}^{N}[y(t) - \hat{y}(t|\theta)]^2 + \frac{1}{2N}\theta^T D\theta \tag{4.44}$$

Matrix D is a diagonal matrix which is often selected as $D = \alpha I$ $(\alpha > 0)$ or $D = 0$, where α denotes weight decay or regularisation parameter that controls the amount of regularisation introduced to the criterion $V_N(\theta, Z_N)$. The larger the α value, the more important the regularisation becomes. The regularisation method prevents the weight from getting larger by minimising the sum square error and the regularisation term.

Apart from augmenting the criterion with regularisation term, the LM algorithm also needs several more modifications to match the regularised criterion by adding additional regularisation terms to the Gradient and Hessian matrix:

$$G(\theta) = W_N'(\theta, Z_N) = \frac{1}{N}\sum_{t=1}^{N}\psi(t|\theta)[y(t) - \hat{y}(t|\theta)] + \frac{1}{N}D\theta \tag{4.45}$$

$$R(\theta) = W_N''(\theta, Z_N) = \frac{1}{N}\sum_{t=1}^{N}\psi(t|\theta)[\psi(t|\theta)]^T + \frac{1}{N}D \tag{4.46}$$

The ratio $r^{(i)}$ for updating the parameter $\lambda^{(i)}$ apparently needs to be changed to:

$$r^{(i)} = \underbrace{\frac{2\left[V_N\left(\theta^{(i)}, Z_N\right) - V_N\left(\theta^{(i)} + f^{(i)}, Z_N\right)\right]}{\left(f^{(i)}\right)^T\left(G\left(\theta^{(i)}\right) + \left[\lambda^{(i)}I + \frac{1}{N}D\right]f^{(i)}\right)}}_{\text{reduction approximation}} \tag{4.47}$$

4.3.5 Recursive based Neural Network Model Estimation

The recursive system identification method builds a model of the system at the same time as the measurement data is collected. The prediction model is then updated at each time step, as new data become available. In our study, the weight updating procedure is calculated using recursive Gauss Newton (rGN) method. It was originally derived by Ljung and Soderstrom [1983] and later on modified in Chen et al. [1990] to train the MLP network. For every data sample, the parameter vector $\hat{\theta}(t)$ is updated by the recursive algorithm using the following equations:

$$e(t) = y(t) - \hat{y}(t) \tag{4.48}$$

$$R(t) = \lambda(t)R(t-1) + (1-\lambda(t))\,\psi(t)\,\hat{\Lambda}^{-1}(t)\,\psi^{T}(t) \tag{4.49}$$

$$K(t) = (1-\lambda(t))\,R^{-1}(t)\,\psi(t)\,\hat{\Lambda}^{-1}(t) \tag{4.50}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + K(t)\,e(t) \tag{4.51}$$

where $R(t)$ is an approximation of the Gauss-Newton Hessian matrix, $\hat{\theta}(t)$ is the estimation of parameters vector of the neural network model, $\hat{\Lambda}^{-1}(t)$ is the weighting matrix and $\lambda(t)$ denotes the forgetting factor at the current time step t. The simplest choice of weighting matrix $\hat{\Lambda}^{-1}(t)$ is an identity matrix as suggested by Billings et al. [1992]. The forgetting factor $\lambda(t)$ is defined as a constant scalar variable which accounts for the amount of past data information to be included in the error criterion function. If the forgetting factor is $\lambda(t) < 1$, the term would make the estimation more adaptable to changes and sensitive to noise. Whereas, if $\lambda(t) \to 1$ as time increases, more old data are included in the criterion and the adaptation would fluctuate less during the learning process [Youmin and Li, 1999].

In practice, Equation (4.48)-(4.51) are not calculated straightforward with inversion of matrix $R^{-1}(t)$ which requires computational complexity of $O(d^3)$ [Ngia and Sjoberg, 2000]. Ljung and Soderstrom [1983] had shown that the matrix inversion complexity can be reduced from complexity of $O(d^3)$ to $O(d^2)$ using matrix inversion theorem as follows:

$$[A+BCD]^{-1} = A^{-1}B\left[DA^{-1}B + C^{-1}\right]^{-1}DA^{-1} \tag{4.52}$$

By applying Equation (4.52) to (4.49) with $A = \lambda(t)R(t-1)$, $B = (1-\lambda(t))\psi(t)$, $C = 1$, $D = \psi^T(t)$ and introducing term $P(t) = (1-\lambda(t))R^{-1}(t)$, the generalised rGN algorithm in Equation (4.48)-(4.51) are rewritten as follows to avoid full Hessian matrix inversion:

$$P(t) = \left[P(t-1) - L(t)S^{-1}(t)L^T(t)\right]/\lambda(t) \tag{4.53}$$

$$S(t) = \psi^T(t)P(t-1)\psi(t) + \lambda(t)\hat{\Lambda}(t) \tag{4.54}$$

$$L(t) = P(t-1)\psi(t)S^{-1}(t) \tag{4.55}$$

$$\hat{\theta}(t) = \hat{\theta}(t-1) + L(t)e(t) \tag{4.56}$$

Note that the inversion of matrix $R^{-1}(t)$ had been reduced from full inversion of $d \times d$ matrix to $S^{-1}(t)$ with $n \times n$ dimension. Note that n denotes the number of outputs predicted in the model.

Equation (4.53) - (4.56) indicates that initial value of $P(0)$ ($d \times d$ matrix) and parameter vector $\hat{\theta}(0)$ need to be supplied by user at the beginning of the iteration. The initial parameter vector $\hat{\theta}(0)$ is usually selected as random values or pre-determined weights resulting from the off-line training. A common choice for $P(0)$ is $P(0) = \rho I$ with ρ being a large positive number (i.e $10^2 \rightarrow 10^4$) indicating little confidence in $\hat{\theta}(0)$. This would cause the estimation to rapidly increase in the transient phase for a short period of time δt [Ljung and Soderstrom, 1983]. As the estimation of parameters is quite poor at the beginning of the iteration, a lower forgetting factor $\lambda(t)$ should be selected at the initial stage for rapid adaptation and approaching unity as the time increases. The following strategies introduced by Ljung and Soderstrom [1983] is used for updating the forgetting factor term:

$$\lambda(t) = \lambda_o\lambda(t-1) + (1-\lambda_o) \tag{4.57}$$

where λ_o and $\lambda(0)$ are design variables. The typical values of λ_o is 0.99 and $\lambda(0)$ is in the range of $0.95 < \lambda(0) < 1$.

According to Ljung and Soderstrom [1983], the recursion of Equation (4.53) is

Table 4.1 Potter's Square Algorithm

a) Initialise $P(0) = Q(0)Q^T(0)$ at time $t = 0$
b) For each time step t, update $Q(t-1)$ by performing step 1-6
1. $f(t) = Q^T(t-1)\psi(t)$
2. $\beta(t) = \lambda(t) + f^T(t) f(t)$
3. $\alpha(t) = 1/\left[\beta(t) + \sqrt{\beta(t)\lambda(t)}\right]$
4. $\bar{L}(t) = Q(t-1) f(t)$
5. $\bar{Q}(t) = \left[Q(t-1) - \alpha(t)\bar{L}(t) f^T(t)\right]/\sqrt{\lambda(t)}$
6. $Q(t) = \frac{\rho_{max}-\rho_{min}}{tr\{\bar{Q}(t)\}}\bar{Q}(t) + \rho_{min} I$
c) Compute parameter vector as:
$\hat{\theta}(t) = \hat{\theta}(t-1) + \bar{L}(t)\left[e(t)/\beta(t)\right]$

numerically unstable due to round-off errors which build up and influence $P(t)$ to become indefinite. The numerical problem involving matrix $P(t)$ can also be corrected using several matrix factorisation techniques such as Potter's square root algorithm, Cholesky decomposition or UD factorisation which in turn gives better numerical properties compared to the straightforward calculation of $P(t)$ [Ljung and Soderstrom, 1983]. The factorisation algorithms roughly required the same amount of computation to update $P(t)$ in Equation (4.53) [Bierman, 1977]. In this work, the Potter's square root algorithm is considered for the problem because of the algorithm's simple implementation.

The Potter's square root algorithm describes matrix $P(t)$ in terms of the following factorisation:

$$P(t) = Q(t) Q^T(t) \tag{4.58}$$

where $Q(t)$ is selected as a square non-singular matrix. The $Q(t)$ matrix is then calculated using the following algorithm in Table 4.1 at each time step. The implementation of recursive Gauss-Newton algorithm for NN based system identification using Potter's factorisation algorithm is given in Figure 4.13. Since the parameter vector $\hat{\theta}(t)$ in recursive algorithms is updated at the same time as the sensor data is collected, the update remains in indefinite loop as long as the stopping conditions are not satisfied.

Noted that the matrix $P(t)$ in Equation (4.53) may happen to be singular or

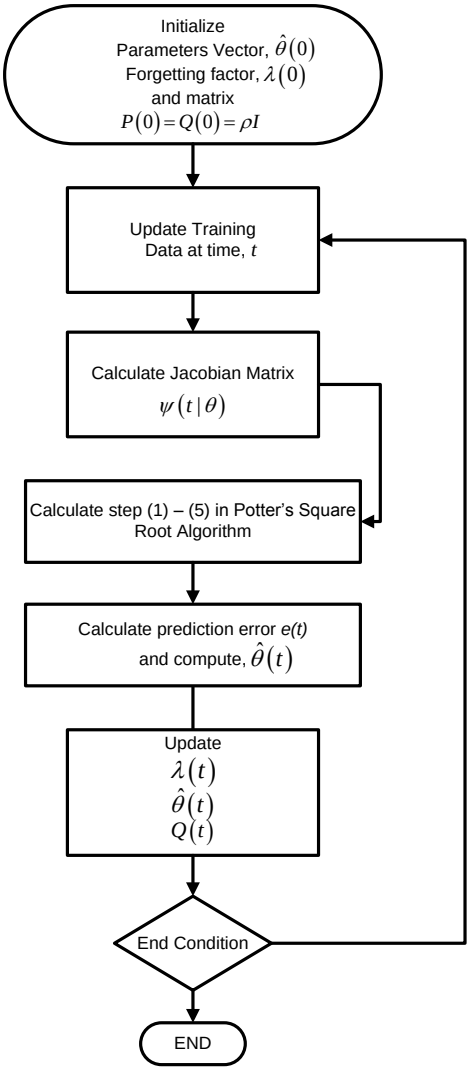


Figure 4.13 The recursive Gauss-Newton (rGN) algorithm with Potter's square root factorisation.

nearly singular if the model set contains too many parameters or if the input signal is not general enough [Ljung and Soderstrom, 1983]. This problem can be overcome by introducing a lower and upper bounds on the eigenvalues of $P(t)$. Several variations of recursive Gauss-Newton algorithms such as Constant Trace (CT) and Exponential Forgetting and Resetting Algorithm (EFRA) have been proposed in various examples to overcome the unstable numerical $P(t)$ recursion [Norgaard, 2000, Salgado et al., 1988]. By using the CT method, Step 6 in Table 4.1 is introduced to bound the eigenvalues of $P(t)$. The ρ_{max} and ρ_{min} are the maximum and minimum eigenvalues respectively, and the values are selected so that $\rho_{max}/\rho_{min} \simeq 10^5$. The initial $Q(0)$ should be selected as

a diagonal matrix, $\rho_{min}I \leq Q(0) \leq \rho_{max}I$.

4.3.6 Model Validation

The model validation process is performed using the second data set $Z_V = [y_V(t), u_V(t)]$ that is different from the training data set. The validation results are based on three analyses: one-step ahead predictions, k-step ahead predictions and k-folds cross validation of the predictions.

One-step ahead predictions are a simple plot that compares the actual measurement data with the model prediction over a test data set. The k-step ahead predictions are normally carried out to detect further deficiency in the fitted model since under high frequency sampling, a one-step ahead visual inspection usually gives a very small prediction error. The calculation example of k-step ahead prediction is shown in Figure 4.14. The k-step ahead prediction is calculated starting from the first step prediction using past output and input data. The predicted outputs from the first step are used as substitutes for the measured output data for the second step prediction since the actual system observations are not available in future predictions. This process continues until the final k-step ahead prediction is obtained. The k-step ahead prediction in a mathematical compact form is given in Norgaard [2000]. The overall accuracy of the prediction error can be represented in scalar quantity according to mean square error (MSE) criterion or root mean square error (RMSE), percentage RMSE and the R^2 criterion as in the following formulation [Suresh et al., 2003, Norgaard, 2000, Shamsudin and Chen, 2012a]:

$$RMSE = \sqrt{\frac{1}{N}\sum_{t=1}^{N}\left(\hat{y}_i(t) - y_i(t)\right)^2} \tag{4.59}$$

$$\%RMSE = \sqrt{\frac{\sum_{t=1}^{N}\left(\hat{y}_i(t) - y_i(t)\right)^2}{\sum_{t=1}^{N}\left(y_i(t) - y_i(t)\right)^2}} \tag{4.60}$$

$$R^2 = 1 - \frac{\sum_{t=1}^{N}\left(\hat{y}_i(t) - y_i(t)\right)^2}{\sum_{t=1}^{N}\left(y_i(t) - y_i(t)\right)^2} \tag{4.61}$$

Cross validation is a statistical method that is normally used in data mining problems to determine the model structure selection and to compare generalisation

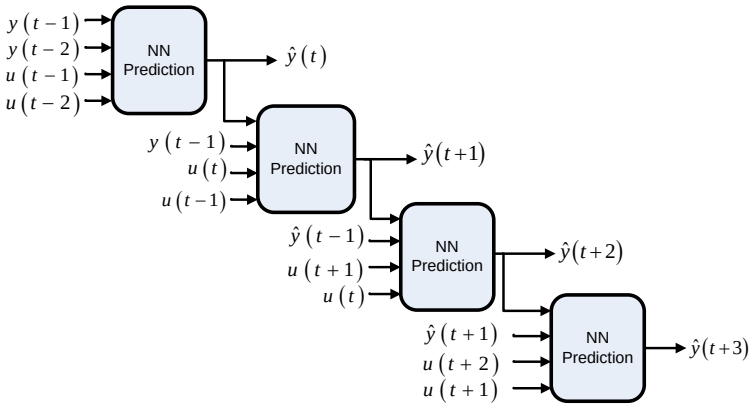


Figure 4.14 The k-step ahead predictions with three step ahead example. Adapted from Samal [2009].

performance of different learning methods. The simplest method to conduct validation analysis is to use the hold-out method where the measurement data is divided into training and test sets with a user defined split ratio. Subsequently, the training set is used for model training and the test set data for error rate estimation of the trained model. However, the downside of this method is that the model evaluation can produces a high variance error. Depending on how the split ratio is defined, the prediction error evaluation may be inconsistent for different partitions of data that forms the training and test sets [Kohavi, 1995].

To overcome this problem and utilising the available overall data, the k-fold cross-validation method was used to reduce the variance by averaging error over k data segments. In this method, the measurement data N is split into k approximately equal size data segments. Then, the training and validation are performed for k-iterations where within each iteration, a single portion of the data segment at a certain index location shown in Figure 4.15 will be used for validation after the training of the remaining $k-1$ data segments are completed. For each validation, the prediction MSE is calculated for the specific segment. The MSEs from each validation segment are averaged and combined together at the end of the iteration process using percentage root mean square error (% RMSE) given by:

$$\%\ RMSE = \left[\frac{N}{kM}\frac{\sum_{i=1}^{k}\sum_{t=1}^{M}(\hat{y}_i(t)-y_i(t))^2}{\sum_{t=1}^{N}(y(t)-\bar{y}(t))^2}\right]^{1/2}\times 100 \tag{4.62}$$

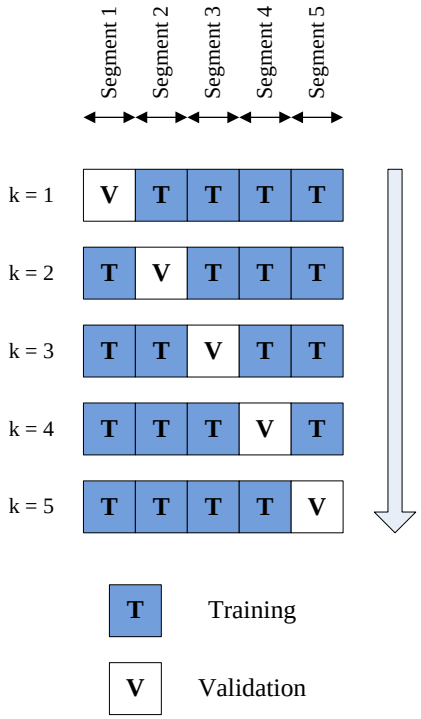


Figure 4.15 The procedure of k-fold cross-validation for $k = 5$.

where $\hat{y}_i(t)$ denotes the predicted NN model output from a specific k-validation data segment, $y_i(t)$ indicates the k-validation data segment, M is the data size in each of the segment and $\bar{y}(t)$ is the mean value of the measurement data.

4.4 SUMMARY

This chapter outlines the basic ANN concepts and proposed various network architectures for the use in NN based system identification. The general guidelines for NN based system identification for modelling the dynamics of the helicopter UAS are presented. The flight data preparations for the off-line system identification are also discussed, followed by detailed explanation of implementation of off-line NN training using LM algorithm. In order to avoid over-fitting problem during training phase, regularisation method with weight decay is introduced to limit the flexibility of the network without relying on early stopping criterion. To overcome the disadvantages of off-line (batch LM) methods, the recursive Gauss-Newton algorithm is used to track the time varying dynamics of the helicopter UAS. The recursive model estimation is a system identification

technique that enables us to infer a model that adapts to time-varying dynamics based on real-time data coming from the system. Finally, the selection methods of neural network model structure are also discussed using the Lipschitz coefficient and model validity tests method. The results and discussion of the proposed NN based identification algorithms and architectures are given in the next chapter.

Chapter 5

NN BASED SYSTEM IDENTIFICATION: RESULTS AND DISCUSSION

5.1 INTRODUCTION

In this chapter, the model selection and validation results from the proposed NN based system identification methods are presented and discussed. The model selection and validation results from the off-line system identification for various NN architectures are presented in Section 5.2, Section 5.3 and Section 5.4. Section 5.5 provides the comparison results between off-line prediction performance from conventional MLP architecture and the proposed HMLP and modified Elman network. In Section 5.6, the identification results using recursive training are presented. The on-line training proposed in Section 4.3.5 is implemented to identify the attitude dynamics of the UAS helicopter model using the MLP network. The prediction performance of the MLP network trained with off-line LM algorithm and the MLP network trained with repeated recursive Gauss-Newton algorithm are compared. The feasibility of implementing recursive type training is analysed against mini-batch LM algorithm in term of training time. Then, the prediction performance comparison of MLP, HMLP and Elman network using the recursive training method is given in Section 5.7. Finally, the chapter is summarised in Section 5.8.

5.2 OFF-LINE BASED SYSTEM IDENTIFICATION FOR MLP NETWORK

In this section, the off-line system identification proposed in Section 4.3.4 is implemented to identify the attitude dynamics of the UAS helicopter model. The identification process is conducted using real flight test data obtained from different flight manoeuvres. The flight test data were divided into training, validation and test data sets. The training and validation data sets were used for purpose of NN training and model structure selection. The test data set was used for the final evaluation of the NN model prediction accuracy and reliability. In each manoeuvre, only a certain input command was used to excite the dynamic of interest. During the experiment, all control inputs and all UAS state measurements were recorded and sampled at 100 Hz. Figure 5.1 shows the recorded data during lateral and longitudinal cyclic swept for 100 s. The data were filtered using a low pass filter at a cut-off frequency 15 Hz to remove the undesirable structural vibrations effect. Using the collected data, the suitable regression vector (network structure) and hidden neurons size were determined using the Lipschitz coefficient and k-fold cross validation technique previously discussed. To avoid over-fitting problem, the adaptability of the NN weights during training was reduced using the regularisation method.

5.2.1 Improving Generalisation of Neural Network through Regularisation

The NN training used in this research work employed regularisation term in the error criterion to improve generalisation performance of the NN model. For a relatively small sized network model, a trial and error approach was adopted to select the appropriate weight decay parameter α to minimise the generalisation error. Figure 5.2 shows the effect of varying weight decay values on the training error, validation error and training iteration using attitude dynamic data set ($y = [p \quad q]^T$; $y = [\delta_{lon} \quad \delta_{lat}]^T$). The result was obtained using the NN model trained with Levenberg-Marquardt (LM) algorithm using 5 hidden neurons and 8 regressors ($n_y = 3$ and $n_u = 1$). As the value of weight

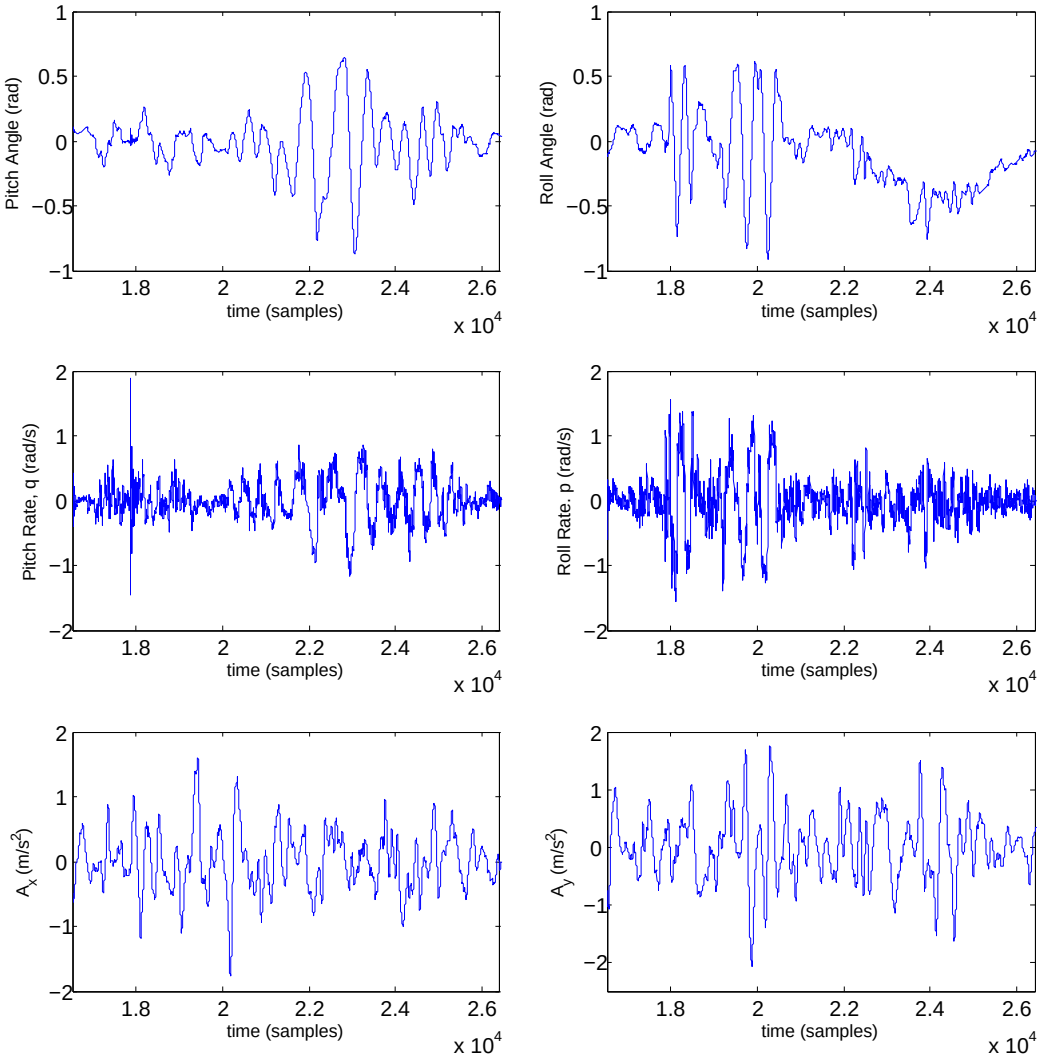


Figure 5.1 Sample of measurement data records during a longitudinal and lateral cyclic swept experiment: (a) The longitudinal (pitch angle θ, pitch rate, q and body acceleration in x-axis, A_x) and lateral (roll angle ϕ, roll rate, p and body acceleration in y-axis, A_y) output plots; and (b) The frequency swept plots of longitudinal cyclic δ_{lon} and lateral cyclic δ_{lat}.

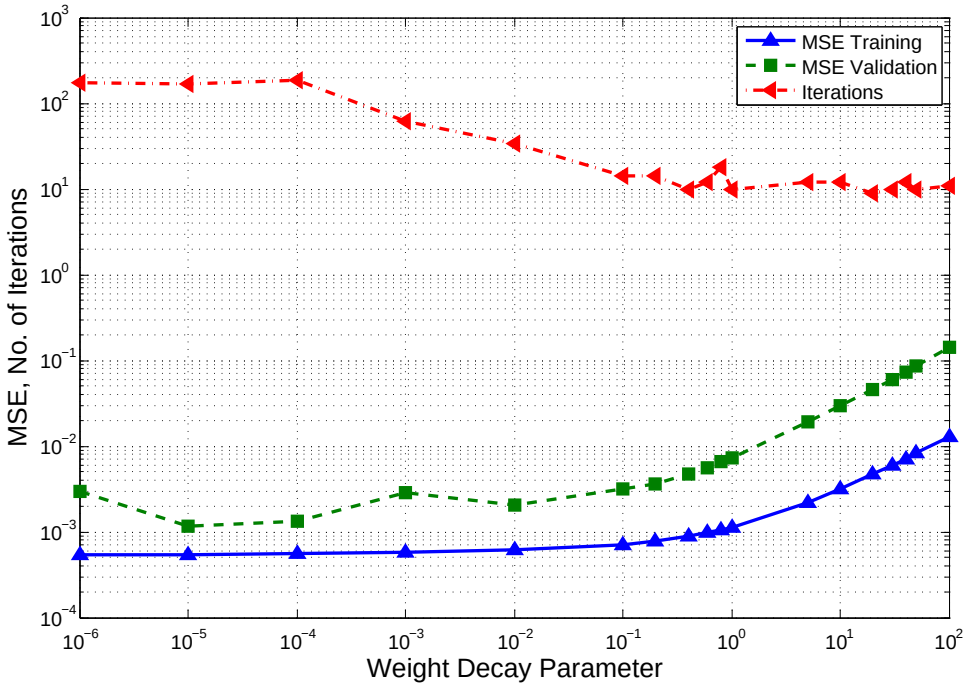


Figure 5.2 The prediction performance result of a MLP network with 5 hidden neurons trained with different weight decay values. The network was trained 5 times using random weight initialisation.

decay parameter α increases, the number of iterations taken to converge to a minimum decreases. This indicates that a higher weight decay parameter α would introduce some early stopping criterion to the training process but at the expense of poor generalisation error. As a trade-off, we could select the value of weight decay parameter α in the range of $10^{-6} < \alpha < 10^{-2}$.

The effect of regularisation term on the network weights adaptation and error evaluation is shown in Figure 5.3. The weights adaptation during training is plotted for every training iteration using previous model structure ($n_y = 3$ and $n_u = 1$). Three different regularisation parameters are used: $\alpha = 0$, $\alpha = 0.0001$ and $\alpha = 0.8$, where $\alpha = 0$ indicates that the training is carried out without the regularisation term. In error evaluation plots, the MSE value for the validation data is calculated in each iteration after obtaining a set of weights from the training process. For general training without regularisation term $\alpha = 0$ (Figure 5.3(b)), the NN training completes at iteration $i = 500$ (pre-selected maximum iteration). However, the MSE calculation from the validation data set points out that the optimum network is obtained at iteration $i = 112$. The NN training should be stopped around this iteration point, further training steps would

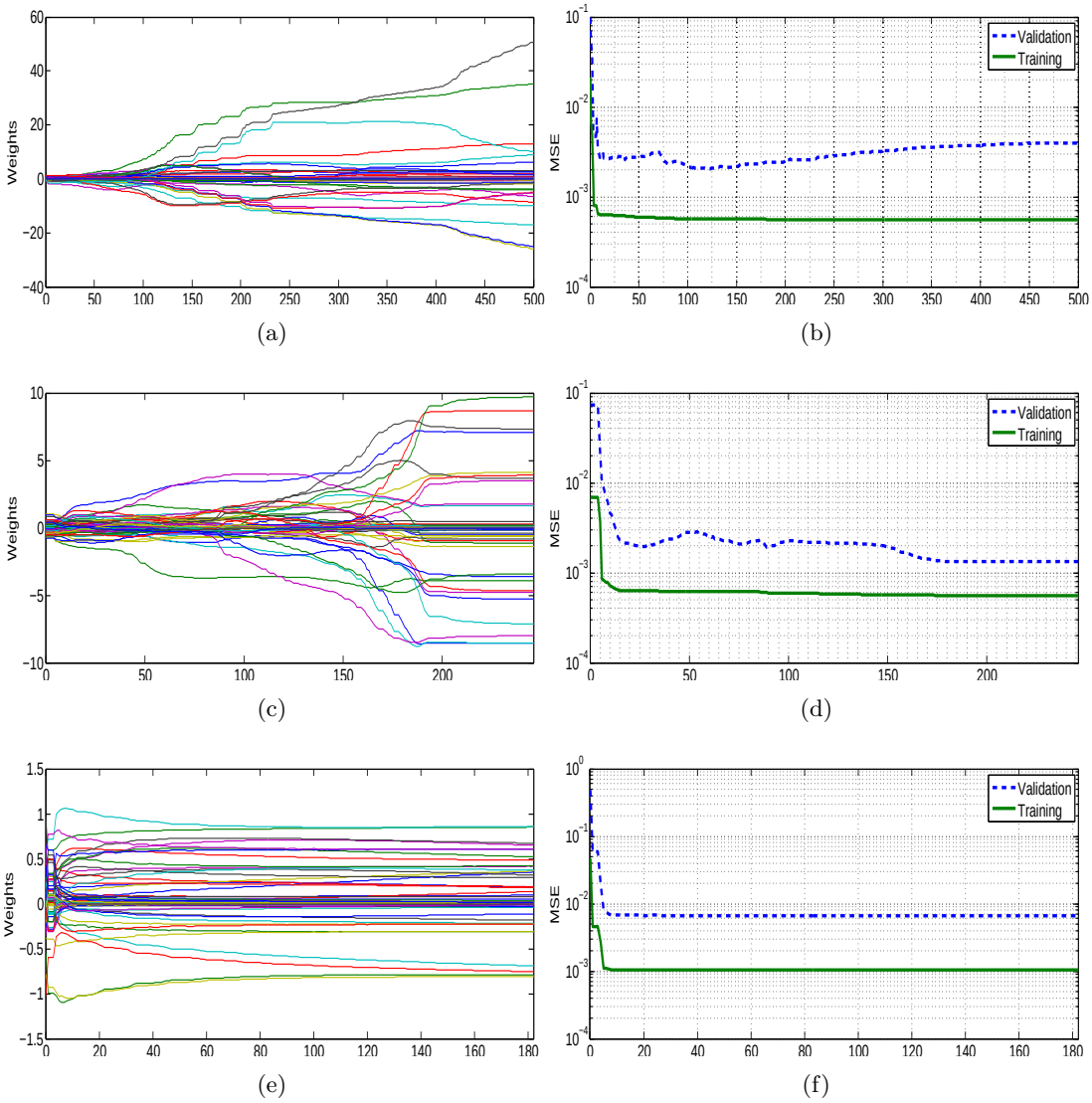


Figure 5.3 The prediction performance of MLP network with varying regularisation parameter α. (a), (c) and (e) show the weights adaptation during the training process where each line represents single weight in the network. The error evaluation on the training and validation datasets is shown in (b), (d) and (f): (a) Network weights adaptation for $\alpha = 0$ (no regularisation term in the error cost function); (b) Training progress for $\alpha = 0$; (c) Network weights adaptation for $\alpha = 0.0001$; (d) Training progress for $\alpha = 0.0001$; (e) Network weights adaptation for $\alpha = 0.8$; and (f) Training progress for $\alpha = 0.8$.

lead to an over-fitting problem as indicated in the MSE plot. The optimum weights are among the smallest at this point compared with those beyond iteration $i = 112$ as shown in Figure 5.3(a). This figure also demonstrates that without regularisation term, some network weights would grow to a large magnitude causing the network to over-fit.

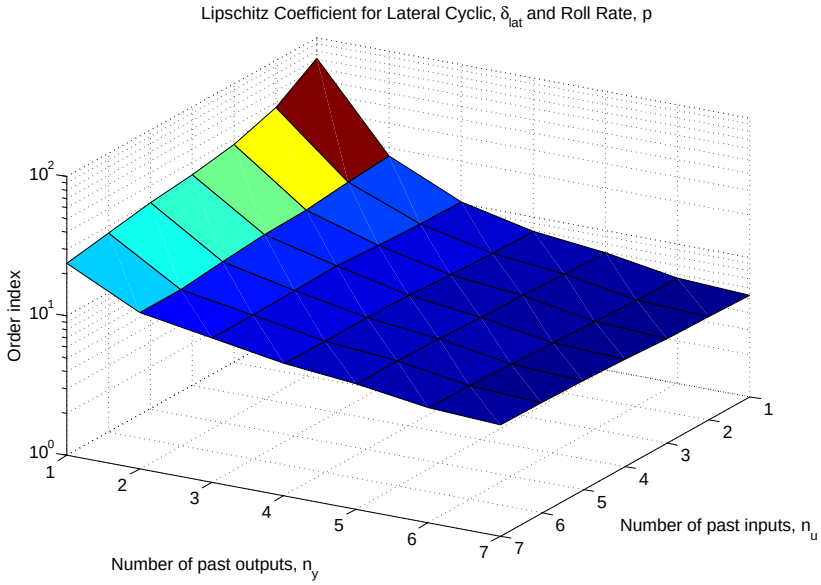
Figure 5.3(d) shows that the training with $\alpha = 0.0001$ completes in 246 iterations, and the optimum weights are found at iteration $i = 185$. The weights adaptation plot

in Figure 5.3(c) shows that the weights gradually increase and remain constant beyond iteration point $i = 185$. The network training ends much quicker compared with the network training without regularisation term since the weights do not change after $i = 185$. The MSE calculation on the validation data set also shows that the optimum weights are found at $i = 185$, thus demonstrating the effectiveness of the regularisation method to prevent the over-fitting problem. The effect of using the regularisation term basically introduces a smoothing effect on the error criterion $V_N(\theta, Z_N)$ in such a way that weights that have less important influence on error are forced to decay towards zero [Samarasinghe, 2007]. In this process, only the important weights that minimise the error are allowed to grow and stabilise at their optimum values.

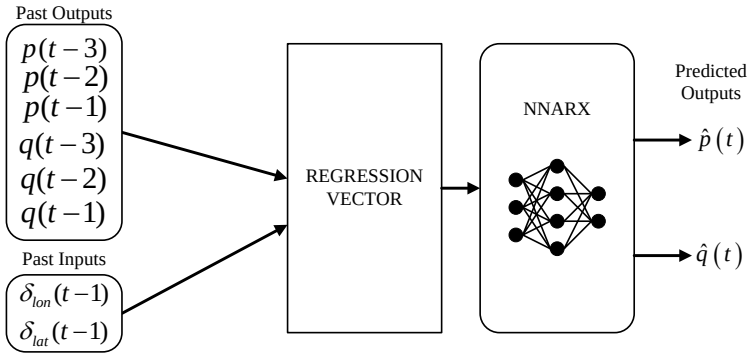
The MSE result for network training with large regularisation parameter $\alpha = 0.8$ is shown in Figure 5.3(f). The figure indicates that the network training stops much earlier than the network training with $\alpha = 0$ and $\alpha = 0.0001$. However, the network model has higher MSE values in both training and validation data. Further increase in regularisation parameter would make the weights less adaptable as the weights only grow in limited small magnitude (± 1.5). This would make the network less flexible and would result in poor prediction performance due to severe bias.

5.2.2 Model Structure Selection Results

The optimum model structure can be found using the Lipschitz coefficient, and it is possible to determine the proper lag space via experimental data [Norgaard, 2000]. The result of the Lipschitz coefficient calculation for a pair of input and output data (δ_{lat} and p) is shown in Figure 5.4(a). It is shown that the Lipschitz coefficient curve decreases and stabilises at $n_y = 3$ and $n_u = 1$ for that particular pair of input and output data selection. For this input-output pair, the reasonable network structure to describe the attitude dynamic ($p/\delta lat$) is by selecting a number of past outputs $n_y = 3$ and number of past inputs $n_u = 1$. By summing all the stabilising points for each data pair, a total number of 8 time regressors are fed to the neural network. Finally, the selected neural network based on the ARX model (NNARX) structure to identify the non-linear relationship of helicopter's attitude dynamics is shown in Figure 5.4(b).



(a)



(b)

Figure 5.4 Preliminary NN model structure selection from experimental input-output data set using the Lipschitz coefficient (a) The Lipschitz coefficient plot obtained for a pair of input and output data; and (b) The NNARX model structure with preselected regression vectors obtained after determining each individual Lipschitz coefficient from respective input-output pair.

The regression vector size can be selected using a much higher number of past outputs and inputs. However, the choice of higher number of past outputs and inputs will result in a larger network architecture that may lower the mean square error (MSE) but with poor generalisation ability [Billings et al., 1992]. This means that the network model would predict the training data set with great accuracy but fail to represent a new data that has not been used in training process.

The method to determine the model order such as the Lipschitz coefficient is known to be sensitive to noise [Sragner and Horvath, 2003]. Normally, if the Lipschitz

coefficient is calculated on noisy data, the coefficient index plot would not exhibit a sharp breakpoint before stabilising at a large model order region. This would lead to incorrect model order selection as network designers would probably select a higher model order in the smooth region. To validate whether NN model structure with 3 past outputs and 1 past inputs represented the correct model structure for the underlying dynamics, the k-fold cross validation was carried out next to determine the best network structure.

Different network structures found from past research works such as Putro et al. [2009] and Samal [2009] were used for comparison to determine the best network structure using k-fold validation method. In this study, the flight data obtained from the experiment was divided into 10 approximately equal segments. In the validation stage, the error calculation was then stored for every network structure and hidden neuron case. Subsequently, the stored error calculation was then retrieved at the end of the validation cycle for RMSE computation.

The result of k-cross validation for different network structures is given in Figure 5.5. Six different network structures were tested and compared with each other. The plot indicates that network structure with 1 past output and 1 past input (4 regressors) gives the highest percentage of RMSE. This indicates that a simple network structure with 1 past output and 1 past input (4 regressors) is not suitable to be used for predicting the non-linear dynamic system. As the number of regressors or inputs to the network increases, the RMSE value decreases and stabilised after 3 past outputs and 1 input (8 regressors) structure. Hence, the neural network model structure can be selected as a total of 8 regressors with 3 past outputs and 1 past input. This cross validation procedure was repeated for different hidden neuron sizes and an overall RMSE trend points to the same sharp breakpoint at 3 past outputs and 1 input (8 regressors) network structure. This further indicates that the generalisation performance of the network is more sensitive to the effect of network structures rather than hidden neurons.

The optimum number of hidden neurons used in the network was also determined using k-fold cross validation method. The simulation result of hidden neuron selection for 3 past outputs and 1 past input network structure is given in Figure 5.6. From the

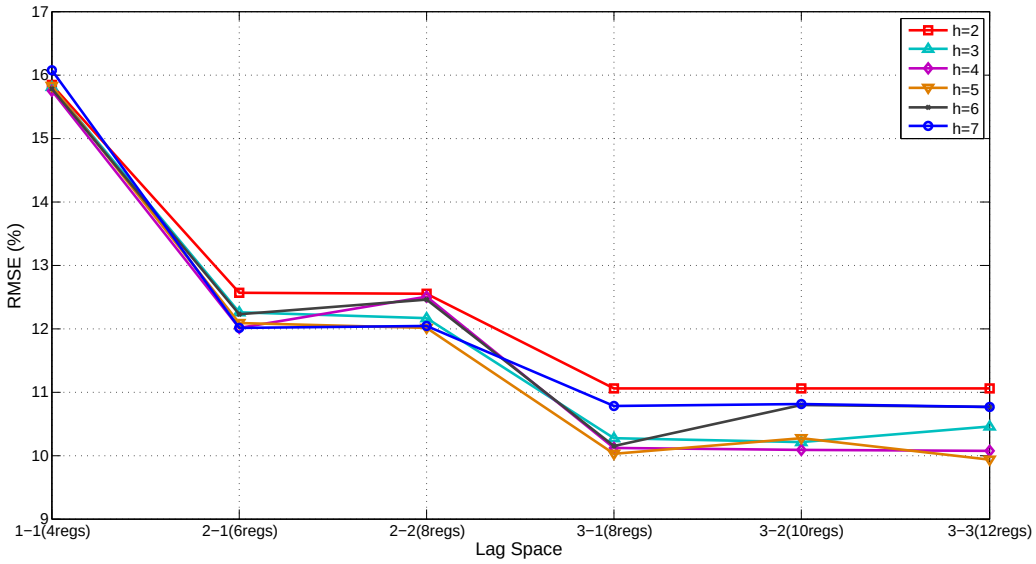


Figure 5.5 The percentage of Root Mean Square Error (RMSE) of MLP network model for each network structure and number of hidden neurons. The neural network training was carried out using off-line Levenberg-Marquardt (LM) algorithm.

plot, network structure with 3 past outputs and 1 past input (regression vector with dimension size of 8) gives the lowest RMSE value for neurons size, $h = 4$. Despite the fact that the neuron size $h = 7 \rightarrow 8$ gives a comparable low RMSE value, it does not indicate that the prediction displays good generalisation performance since neuron size $h = 5$, has a sudden increase in RMSE value. The noise has affected the error calculation for the neuron size $h = 7 \rightarrow 8$. However, the validity test is still useful in aiding the selection of an appropriate network structure [Billings et al., 1992]. Furthermore, it is not advisable to use an excessive number of neurons which may lead to an over-fitting problem. Finally, we arrive at the following network specifications (Table 5.1) that adequately represent the attitude dynamics of a model scaled helicopter.

Table 5.1 The MLP neural networks model parameters.

MLP Network Specifications for Attitude Dynamics	
Number of past outputs	3
Number of past inputs	1
Number of neurons in hidden layer	4
Activation function at hidden layer	Tanh
Activation function at output layer	Linear
Number of regressors	8
Total number of weights	46
Weight decay	0.0001

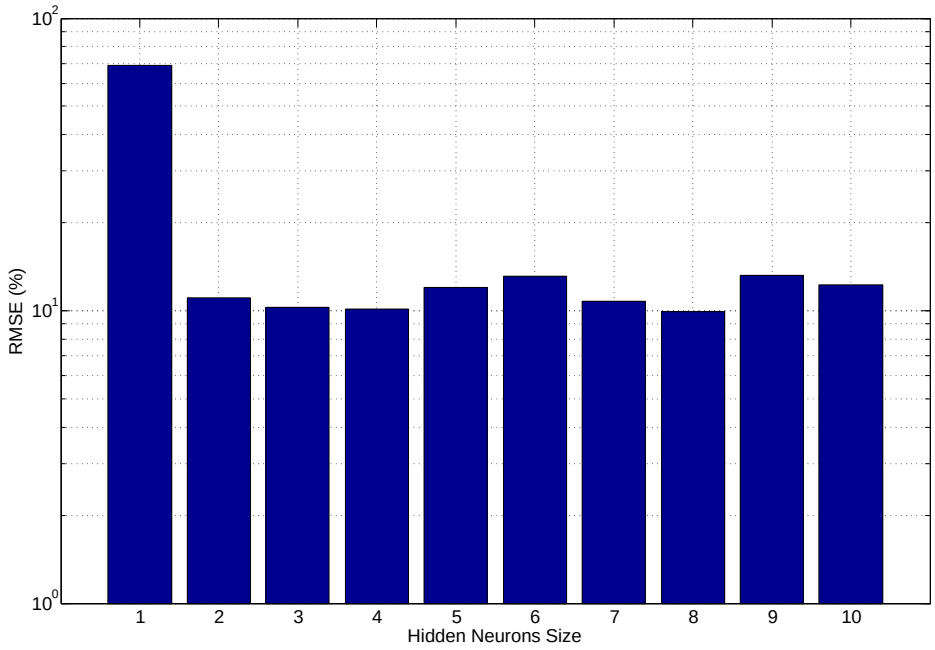


Figure 5.6 The percentage of Root Mean Square Error (RMSE) comparison of MLP network trained with different hidden neuron sizes. The k-cross validation process was conducted for network structure with 8 regressors ($n_y = 3$ and $n_u = 1$). The neural network training was carried out using off-line Levenberg-Marquardt (LM) algorithm.

An example of the one-step ahead prediction of the angular rate responses that are estimated from the off-line neural network system identification is shown in Figure 5.7 and Figure 5.8. The off-line LM training is carried out using the training data set from 16 s to 28 s recording, and the prediction performance is validated on roll rate and pitch rate data (test data set) from 36 s to 37 s. The network is trained using the nearly optimal structure from Table 5.1. These predicted responses from neural network identification (NNID) are overlaid with the measured helicopter responses. The results indicate that one-step ahead NNID predictions overlap the test data almost perfectly as indicated by the magnitude order of the prediction error plot. This usually happens when the sampling frequency of the data collected is high compared with the frequency of the dynamic system as suggested in Norgaard [2000]. The small prediction error from one-step ahead prediction does not necessarily indicate that the model is sufficient without further checks.

The quality of the fitted model is further inspected by running k-step ahead prediction to check if there is a possibility of a significant discrepancy between prediction

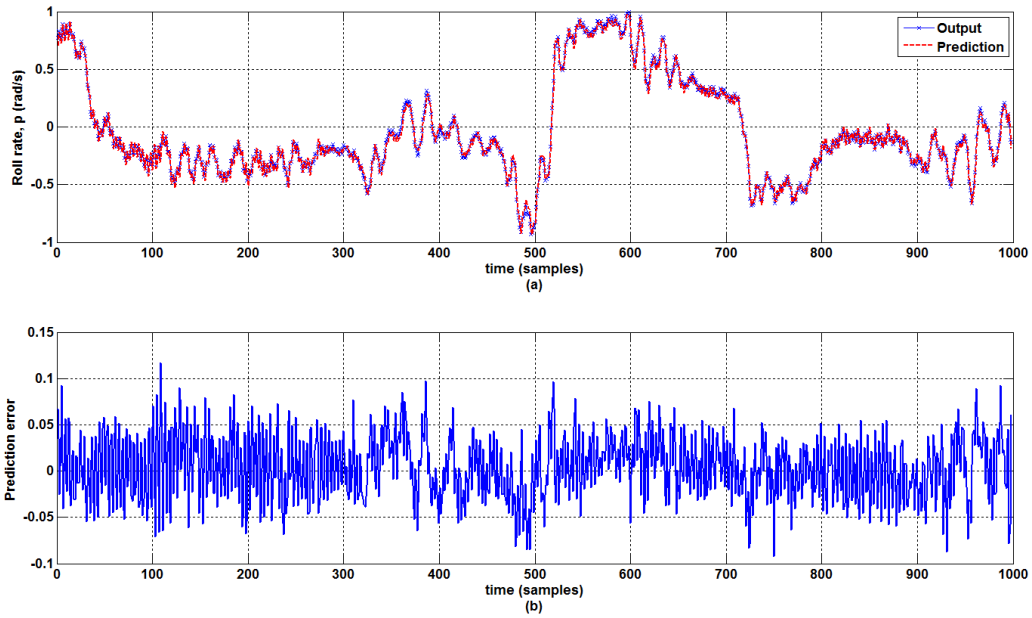


Figure 5.7 The prediction from MLP network model for roll dynamics. (a) The one-step ahead prediction and measurement data plot. (b) The error plot between one-step ahead prediction and the measurement data. The red dashed line indicates estimation from the neural network model while the solid blue line with 'x' marker represents the output measurement.

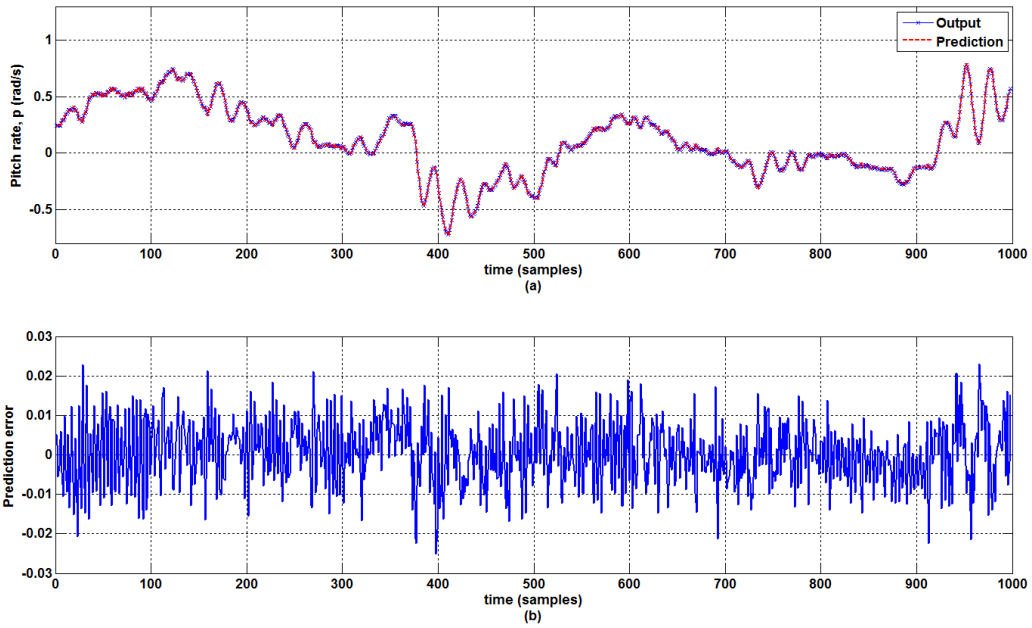
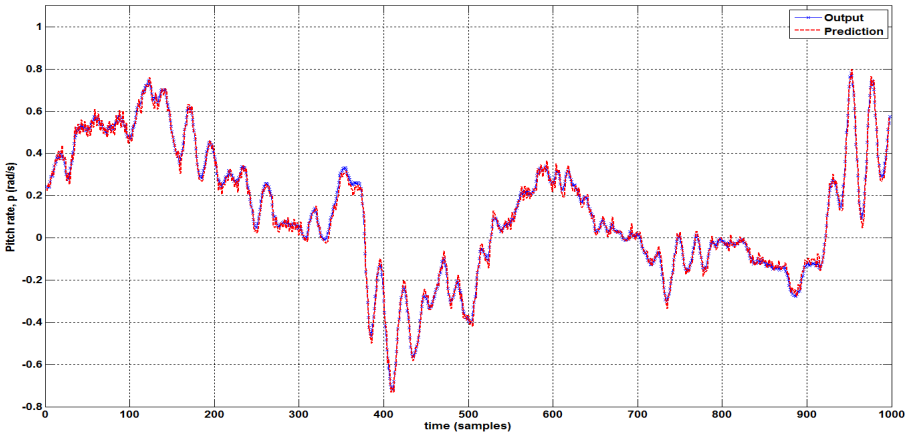
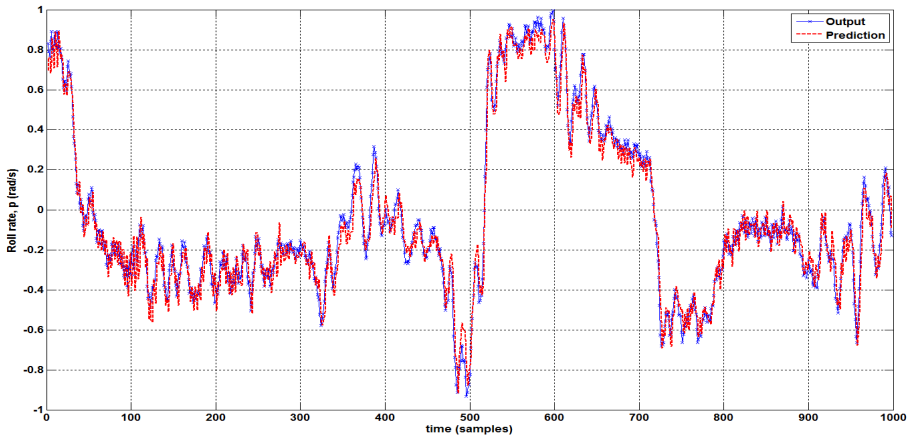


Figure 5.8 The prediction from MLP network model for pitch dynamics. (a) The one-step ahead prediction and measurement data plot. (b) The error plot between one-step ahead prediction and the measurement data. The red dashed line indicates estimation from the neural network model while the solid blue line with 'x' marker represents the output measurement.



(a)



(b)

Figure 5.9 The 5 steps ahead prediction of MLP network model for (a) Pitch dynamics; and (b) Roll dynamics. The solid blue line with 'x' marker is the measured output while the red dashed line indicates the prediction from neural network model.

and measured output data. The plots of the k-steps predictions are given in 5.9(a) and 5.9(b). The corresponding error statistics of one step and k-step ahead predictions are given in Table 5.2 for each individual prediction variable. From the error statistics, we can conclude that the discrepancy between the k-step ahead prediction ($k = 5$) and the measured data is insignificant. Both figures show that the trained neural network model predictions are close to the measured values and that the neural network is properly trained to mimic the rigid body dynamics of the helicopter.

The MLP based NN model produced different magnitude of weight values if the NN training was conducted several times using random initial weight values at the beginning of the training. This would result in many possible sets of weights that would

Table 5.2 Summaries of System Identification Error Statistics for MLP network model.

Test Error Statistics			
System Responses	RMSE	RMSE (%)	R^2
One-Step Ahead Prediction			
p	0.0351	8.4011	0.9929
q	0.0081	2.7038	0.9993
5-Step Ahead Prediction			
p	0.0746	17.8684	0.9681
q	0.0232	7.7531	0.9940

achieve the desired prediction response. The optimal network structure determined from k-fold cross validation method would reduce the redundant weight, and pin-point the nearly optimal number of weights in the network. However, random noise in the training process could cause the weights to fluctuate according to Samarasinghe [2007]. It is important for us to prove that the network predictions are robust against the fluctuation of weights. The network prediction performance is analysed by adding the effect of random noises with increasing magnitude to the weights obtained from the optimal network structure.

Table 5.3 shows the prediction results of optimal network structure for MLP network over test data set with addition of Gaussian distributed random noise to the optimal weights. The optimal weights are corrupted by random noise with zero mean and standard deviation s of 0.01, 0.05, 0.1 and 0.2. For each noise level, 300 sets of weights around the optimum weights are generated, which results in average RMSE and R^2 in Table 5.3. The average RMSE on the test data set for various noise levels indicates that an exceptional prediction performance is achieved up until $s = 0.1$ random noise added to the optimal weight.

Using the 300 set of weights generated, the 95% confidence intervals can be constructed for the optimal weights using the standard statistical inference method $(\bar{w} \pm t_{\alpha,n-1}\sigma_w/\sqrt{N})$. Parameter $\bar{w}$ is the mean of a weight, σ_w is the standard deviation of that weight and N is the number of samples in the test set. The $t_{\alpha,n-1}$ is the t value from t-distribution for $(1-\alpha)$ confidence level with degree of freedom of $N-1$. Hence, the upper and lower limit of the network output $\hat{y}$ performance can be constructed

using the upper and lower confidence intervals (CI) limit of the optimal weights. The measure of upper U_i and lower L_i limit width of the network output is given by taking the average of interval range over the measurement sample [Khosravi et al., 2011]. This is also known as Normalised Mean CI width (NMCIW) and can be used to show the variation of the targets. The measure formulation is given as follows:

$$NMCIW = \frac{1}{N}\sum_{i=1}^{NR}(U_i - L_i) \tag{5.1}$$

The average range of the upper and lower output performance (NMCIW) for each noise level cases are given in Table 5.3. As can be seen from the result, the weights set added random noise with $s = 0.2$ provide a wide range of confidence interval (28.3546%) compared with the range of the measurement between -1 rad/s to 1 rad/s. This indicates that the prediction is imprecise and unreliable to produce predictions that represent the real target values. In this study, a 20% threshold value is used to determine whether the CI is too wide or not.

Table 5.3 The average RMSE for various noise levels applied to optimum weights of MLP network (4 hidden neurons with 3 past outputs and 1 past input).

Test Error Statistics					
Standard Deviation of Noise	System Responses	RMSE	RMSE (%)	R2	NMCIW (%)
0.01	p	0.0375	8.9256	0.9919	0.4232
	q	0.0082	2.5568	0.9992	1.4598
0.05	p	0.0391	9.3019	0.9913	2.0441
	q	0.0134	4.1912	0.9980	7.2532
0.1	p	0.0683	16.2613	0.9733	4.4477
	q	0.0365	11.3787	0.9851	14.7242
0.2	p	0.1779	42.3673	0.8185	8.8449
	q	0.0677	21.1145	0.9488	28.3546

5.3 OFF-LINE BASED SYSTEM IDENTIFICATION FOR HMLP NETWORK

In this section, the results of the model structure and hidden neurons size selection of HMLP network are presented. The identification of UAS helicopter attitude dynamics was carried out using the off-line Levenberg-Marquardt training as in Section 4.3.4. The flight data sets was obtained by performing different flight manoeuvres to excite the dynamic of interest. Using the collected data, the suitable regression vector (network structure) and hidden neurons size were determined using the k-fold cross validation technique as previously discussed. The method for minimising the over-fitting effect in the HMLP network training is similar to regularisation method used in MLP network.

The results of cross validation for HMLP network with different network structures (input nodes) are given in Figure 5.10. Six different network structures were tested and compared with each other. The plot indicates that the simple HMLP network structure with 1 past output and 1 past input (4 regressors) gives the highest percentage of RMSE, which is similar to cross validation trend in MLP, and it is not fit for predicting the non-linear dynamics of the helicopter UAS. As the number of regressors or inputs to the network increases, the RMSE value decreases and stabilises after 2 past outputs and 1 input (6 regressors) structure. Hence, the neural network model structure can be selected as a total of 6 regressors with 2 past output and 1 past input observation. This cross validation procedure was repeated for different hidden neuron sizes and an overall RMSE trend points out to the same sharp breakpoint at 2 past outputs and 1 input (6 regressors) network structure.

The result of neurons size selection is reproduced here for the HMLP network using the k-fold cross validation method. The result of the hidden neurons selection for 2 past outputs and 1 past input (6 regressors) network structure is given in Figure 5.11. From the plot, the network structure with 2 past outputs and 1 past input (regression vector with dimension size of 6) yield the lowest RMSE value (9.82 %) for neurons size, $h = 3$. Finally, we arrive at the following network specifications (Table 5.4) that adequately represent the attitude dynamics of a model scaled helicopter. By examining the RMSE

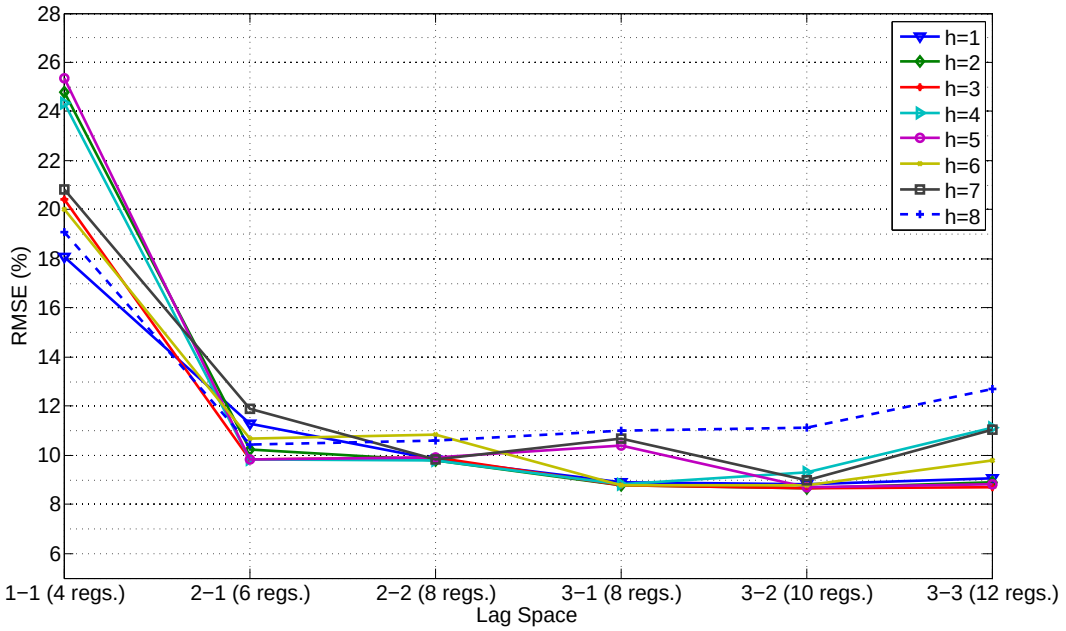


Figure 5.10 The percentage of Root Mean Square Error (RMSE) of the HMLP network trained with different network structures and number of hidden neurons. The neural network training was carried out using off-line Levenberg-Marquardt (LM) algorithm.

Table 5.4 The HMLP neural networks model parameters.

HMLP Network Specifications for Attitude Dynamics	
Number of past outputs	2
Number of past inputs	1
Number of neurons in hidden layer	3
Activation function at hidden layer	Tanh
Activation function at output layer	Linear
Number of regressors	6
Total number of weights	43
Weight decay	0.0001

plots of the optimum HMLP structure, the network is found capable of producing RMSE as good as the standard MLP network at much lower number of hidden neurons and network structure. This suggests that the additional linear connections from the input layer to output layer in the HMLP network helps reduces the complexity of the MLP network by incorporating linear weights connections instead of adding more neurons in the hidden layer. Furthermore, the signal propagations in the linear weight connections across layers are easier to train compared with signals that pass through non-linear neurons [Wilamowski, 2009]. Thus, the reduced network complexity in the HMLP network can leads to a faster learning rate such as demonstrated in Section 5.7.

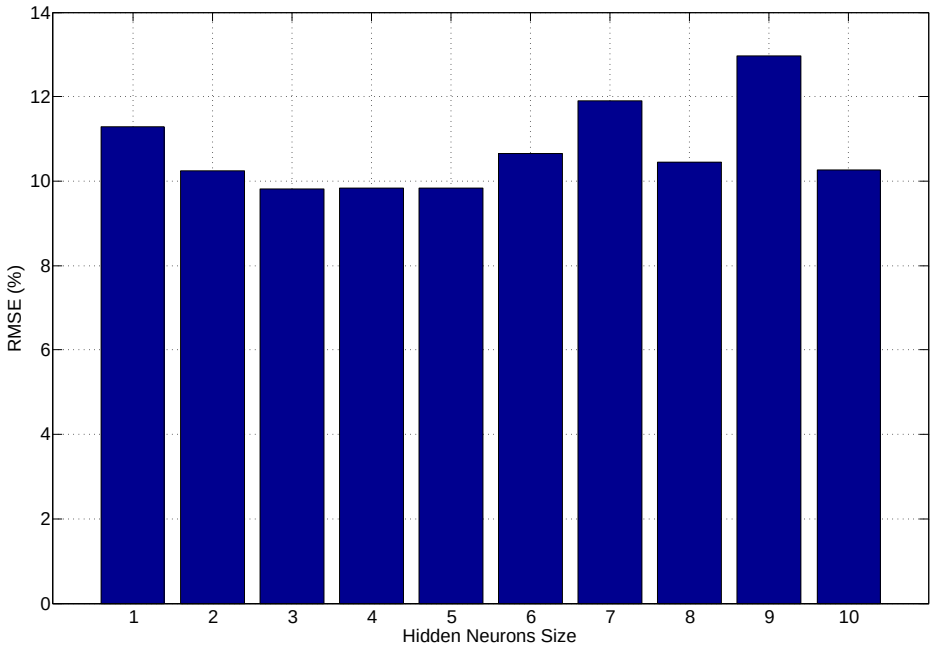


Figure 5.11 The percentage of Root Mean Square Error (RMSE) comparison for different hidden neurons selection. The k-cross validation process was conducted for HMLP network with network structure of 6 regressors ($n_y = 2$ and $n_u = 1$). The neural network training was carried out using off-line Levenberg-Marquardt (LM) algorithm.

The corresponding one-step ahead prediction of the angular rate responses that are estimated from the off-line HMLP neural network model is shown in Figure 5.12 and Figure 5.13. The off-line LM training is carried out using training data set from 16 s to 28 s, and the prediction performance is validated on roll rate and pitch rate data (test data set) from 36 s to 37 s. The network is trained using the nearly optimal structure from Table 5.4. These predicted responses from HMLP network are overlaid with the measured helicopter responses from the test data set. The results indicate that one-step ahead HMLP network prediction overlaps the test data almost perfectly as indicated by the magnitude order of the prediction error plot. Again, this usually happens due to the effect of high sampling frequency of the data collected.

The corresponding error statistics of one-step and k-step ahead predictions are given in Table 5.5. From the error statistics, we can conclude that the discrepancy between the one step or k-step ahead prediction ($k = 5$) and the measured data is insignificant. Overall, we can conclude that the trained HMLP neural network model predictions are close to the measured values and that the neural network is properly trained to mimic the rigid body dynamics of the helicopter.

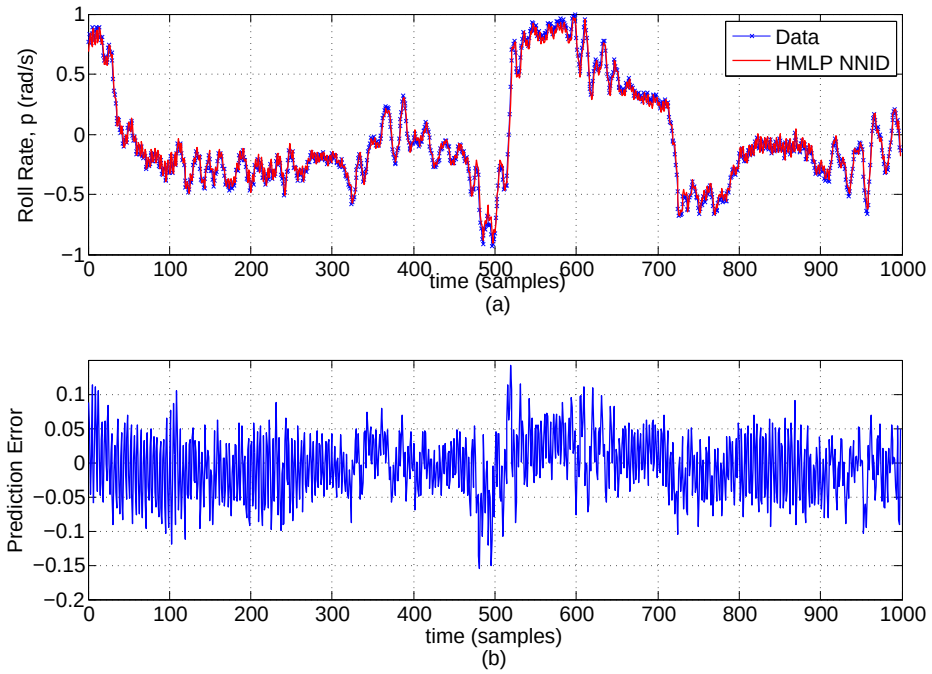


Figure 5.12 The prediction from the HMLP network for roll dynamics. (a) The one-step ahead prediction overlaid with the measured helicopter responses. (b) The error plot between one-step ahead prediction and the measurement data. The red dashed line indicates estimation from the HMLP neural network model while solid blue line with 'x' marker represents the output measurement.

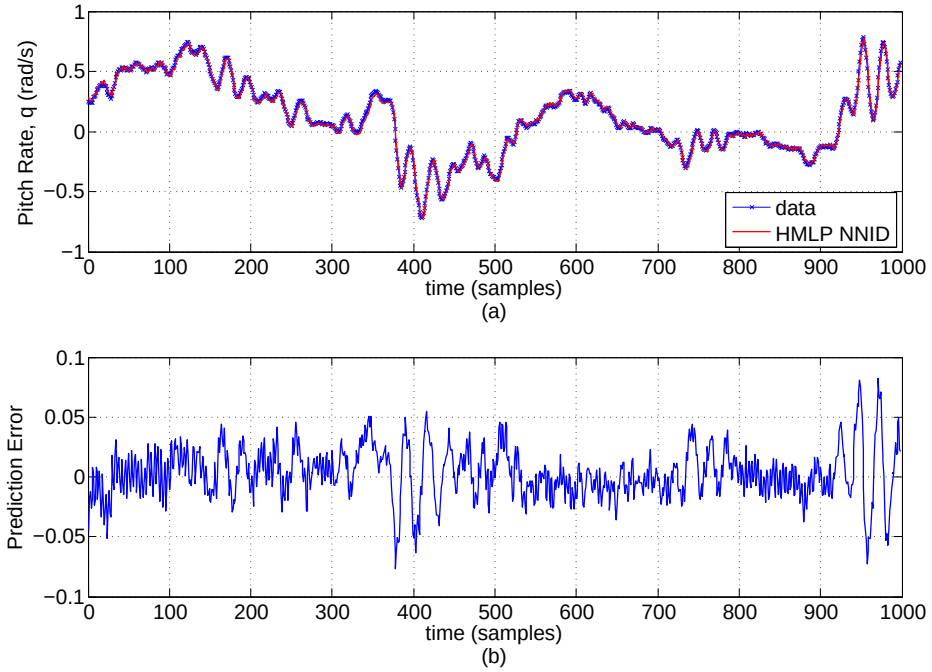


Figure 5.13 The prediction from HMLP network for pitch dynamics. (a) The one-step ahead prediction overlaid with the measured helicopter responses. (b) The error plot between one-step ahead prediction and the measurement data. The red dashed line indicates estimation from the HMLP neural network model while solid blue line with 'x' marker represents the output measurement.

Table 5.5 Summaries of Error Statistics of HMLP Network Model

Error Statistics			
System Responses	RMSE	RMSE (%)	R^2
One-Step Ahead Prediction			
p	0.0360	12.0944	0.9852
q	0.0082	3.6340	0.9984
5-Step Ahead Prediction			
p	0.1199	28.7239	0.9175
q	0.0718	23.9955	0.9424

The robustness of the HMLP network's model structure against perturbation of weights is given in Table 5.6. Table 5.6 shows the prediction results of optimal network structure for HMLP network with addition of Gaussian distributed random noise to the optimal weights. The optimal weights is corrupted by random noise with zero mean and standard deviation s of 0.01, 0.1, 0.3, 0.5, 0.7 and 0.9. For each noise levels, 300 sets of weights around the optimum weights are generated, which results in average RMSE and R^2 values shown in Table 5.6. The average RMSE on the test data set for various noise levels indicate that an exceptional prediction performance is achieved up until random noise with standard deviation $s = 0.7$ added to the optimal weights.

Using the 300 set of weights generated, the 95% confidence intervals can be constructed for the optimal weights using the standard statistical inference method. The average range of the upper and lower output performance (NMCIW) for each noise level case is also given in Table 5.6. As can be seen from the result, the weights set added with $s = 0.9$ random noise provide a wide average confidence interval (23.9983%) compared with the range of measurement between -1 rad/s to 1 rad/s. This indicates the imprecise and high level of uncertainty to produce predictions that represent the real output values.

Table 5.6 The average RMSE for various noise levels applied to optimum weights of HMLP network (3 hidden neurons with 2 past outputs and 1 past input).

Validation Error Statistics					
Standard Deviation of Noise	System Responses	RMSE	RMSE (%)	R2	NMCIW (%)
0.01	p	0.0483	11.4856	0.9867	0.1480
	q	0.0120	3.7572	0.9984	0.2686
0.1	p	0.0469	11.1659	0.9873	1.6318
	q	0.0125	3.9098	0.9982	2.6325
0.3	p	0.0563	13.3997	0.9819	4.7596
	q	0.0576	17.9818	0.9629	8.6277
0.5	p	0.0475	11.2901	0.9871	8.3209
	q	0.0478	14.9172	0.9744	14.7704
0.7	p	0.0596	14.1749	0.9797	11.2747
	q	0.0674	21.0122	0.9493	18.3889
0.9	p	0.2141	50.9196	0.7379	15.1188
	q	0.1942	60.5657	0.5786	23.9983

5.4 OFF-LINE BASED SYSTEM IDENTIFICATION FOR ELMAN NETWORK

The system identification results for the modified Elman network are presented in this section. Similarly to the previous section, the modified Elman network architecture is used to identify the UAS helicopter attitude dynamics using off-line Levenberg-Marquardt training. Since the Elman network only uses the current measurement data to feed into the network, it is not necessary for us to predetermine the appropriate regression vector (network structure) before the NN training. Instead, the strength of self connection α in the network and the effect of hidden neuron sizes are the main factors to consider for ensuring a good generalisation performance. This could be done through the usage of k-fold cross validation technique for hidden neuron sizes selection. For selection of self connection α strength, a simple trial error approach was employed to determine the best gain for network's memory capacity. To avoid over-fitting during training phase, the adaptability of the NN model while training is reduced with similar regularisation method as in the MLP and HMLP network.

The effect of self connection α strength effect on the validation RMSE is shown in Figure 5.14. The simulations were conducted with varying strength of self connection α

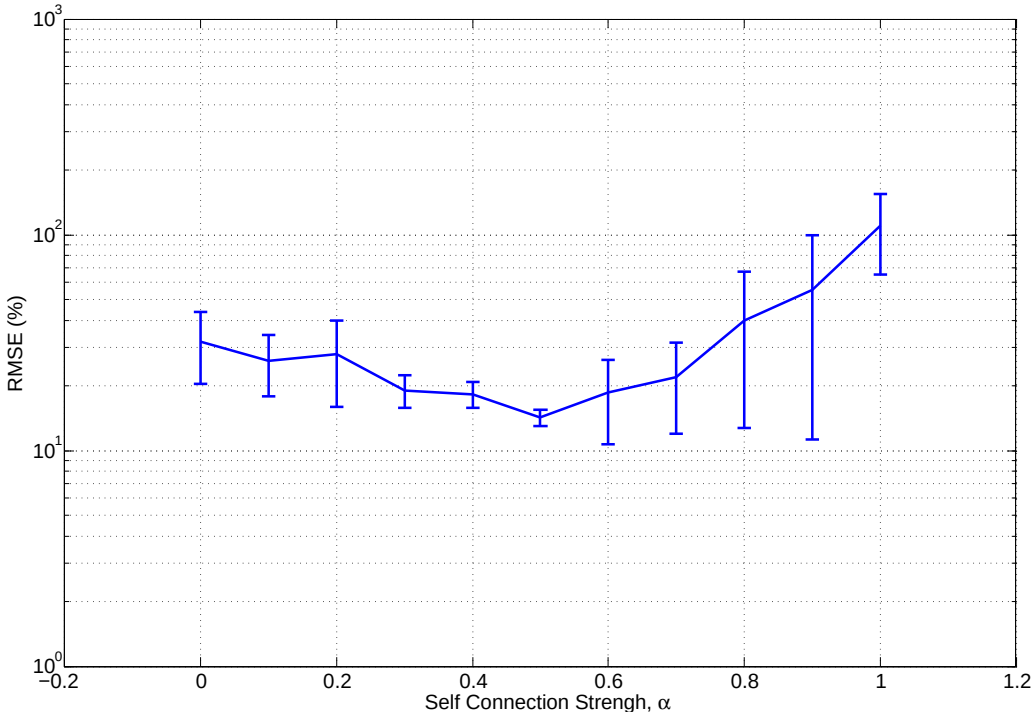


Figure 5.14 The self connection α strength selection results using modified Elman network with reduced connection. The Elman network training are repeated 10 times and validated on a test set. The training is carried out using 6 hidden neurons.

from 0 to 1 to investigate the prediction performance of basic Elman network ($\alpha = 0$) and modified Elman network. Finding from the plot shows that the self connection values between 0.3 to 0.5 produced satisfactory prediction performance before RMSE values increase beyond $\alpha = 0.6$. The lowest RMSE value is found at $\alpha = 0.5$ with $14.24 \pm 1.26\%$ and this is used throughout the analysis.

The result of neurons size selection is reproduced here for the modified Elman network using the k-fold cross validation method. The result of the hidden neurons selection for 1 past output and 1 past input (4 regressors) network structure is given in Figure 5.15. From the plot, the network structure with 1 past input and 1 past output measurement produces an acceptable RMSE percentage between hidden neuron sizes $h = 4 \rightarrow 10$, thus it is logical to select the optimum neuron size at $h = 4$. Finally, we arrive at the following network specifications (Table 5.7) that adequately represent the attitude dynamics of a model scaled helicopter. Note that the total number of weights in the optimum Elman network is lesser than the optimum MLP and HMLP network.

The corresponding one-step ahead prediction of the angular rate responses that are

Table 5.7 The modified Elman network parameters.

Elman Network Specifications for Attitude Dynamics	
Number of past outputs	1
Number of past inputs	1
Number of neurons	4
Activation function at hidden layer	Tanh
Activation function at output layer	Linear
Number of regressors	4
Total number of weights	34
Weight decay	0.0001
Self connection strength α	0.5

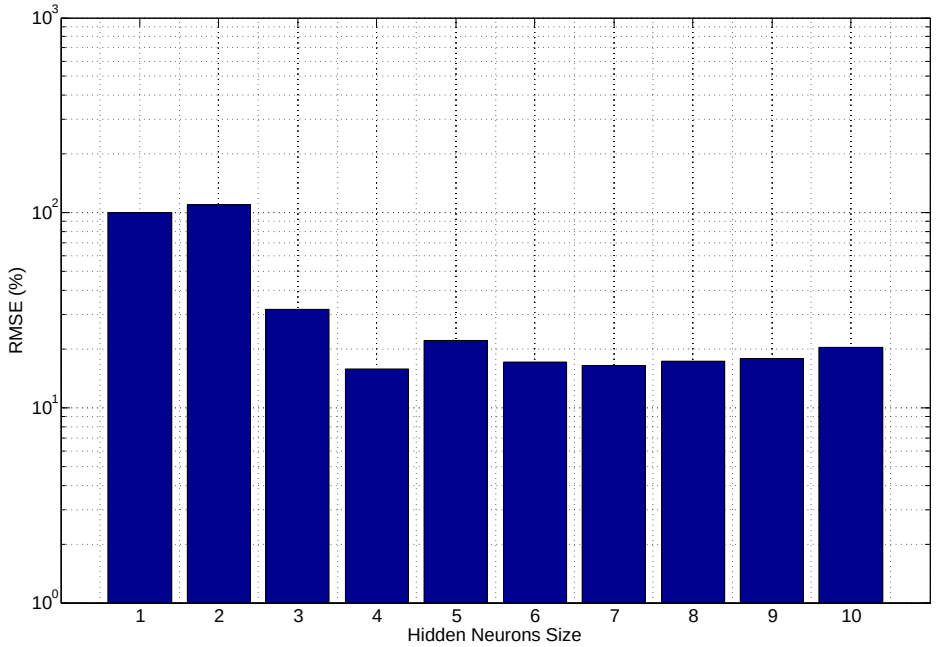


Figure 5.15 The validation RMSE comparison for different hidden neuron sizes. The k-cross validation process was conducted for modified Elman network with network structure of 4 regressors ($n_y = 1$ and $n_u = 1$). The neural network training was carried out using the off-line Levenberg-Marquardt (LM) algorithm.

estimated from the off-line trained modified Elman network is shown in Figure 5.16 and Figure 5.17. The off-line LM training is carried out using training data set from 16 s to 28 s, and the prediction performance is validated on roll rate and pitch rate data (test data) from 36 s to 37 s. The network is trained using the nearly optimal structure from Table 5.7. These predicted responses from modified Elman network are overlaid with the measured helicopter responses from the test data set. The results indicate that one-step ahead modified Elman network prediction overlap the test data almost

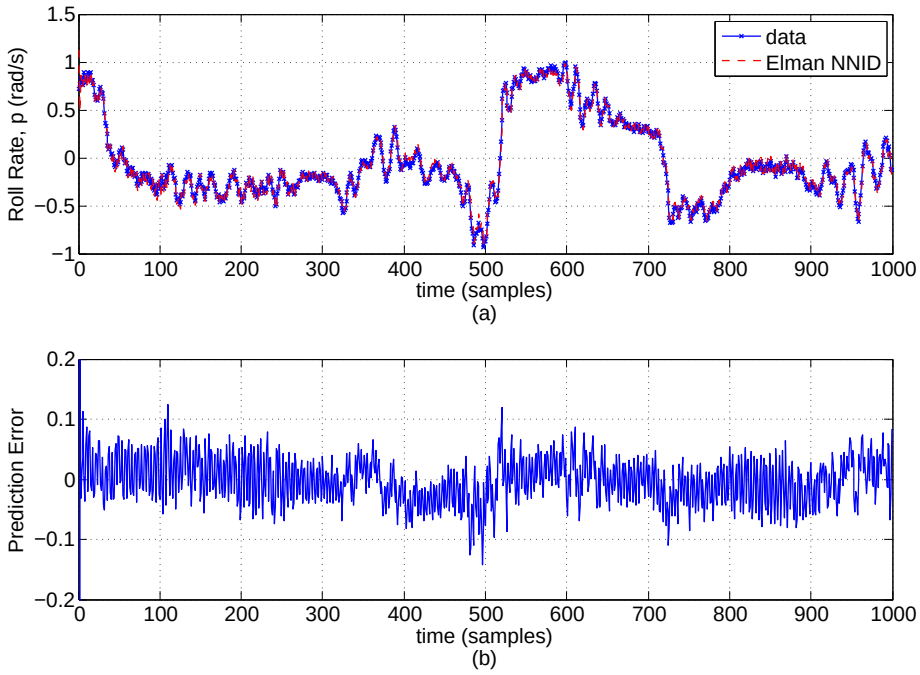


Figure 5.16 The prediction from the modified Elman network for roll dynamics. (a) The one-step ahead prediction overlaid with the measured helicopter responses. (b) The error plot between one-step ahead prediction and the measurement data. The red dashed line indicates estimation from the modified Elman network while solid blue line with 'x' marker represents the output measurement.

perfectly as indicated by the magnitude order of the prediction error plot. Again, this usually happens due to the effect of high sampling frequency of the data collected.

The corresponding error statistics of one-step and k-step ahead predictions are given in Table 5.8. From the error statistics, we can conclude that the discrepancy between the one step or k-step ahead prediction ($k = 5$) and the measured data is insignificant. Overall, we can conclude that prediction from the off-line trained modified Elman network is close to the measured values and that the neural network is properly trained to mimic the rigid body dynamics of the helicopter.

The robustness of the modified Elman network's model structure against perturbation of weights is given in Table 5.9. Table 5.9 shows the prediction results of optimal network structure for the modified Elman network with addition of Gaussian distributed random noise to the optimal weights. The random noise value is adjusted with zero mean and standard deviation s of 0.01, 0.05, 0.1, 0.3, 0.5 and 0.7. For each noise level, 300 sets of weights around the optimum weights are generated, which gives the average RMSE and R^2 values shown in Table 5.9. The average RMSE on the test data set for

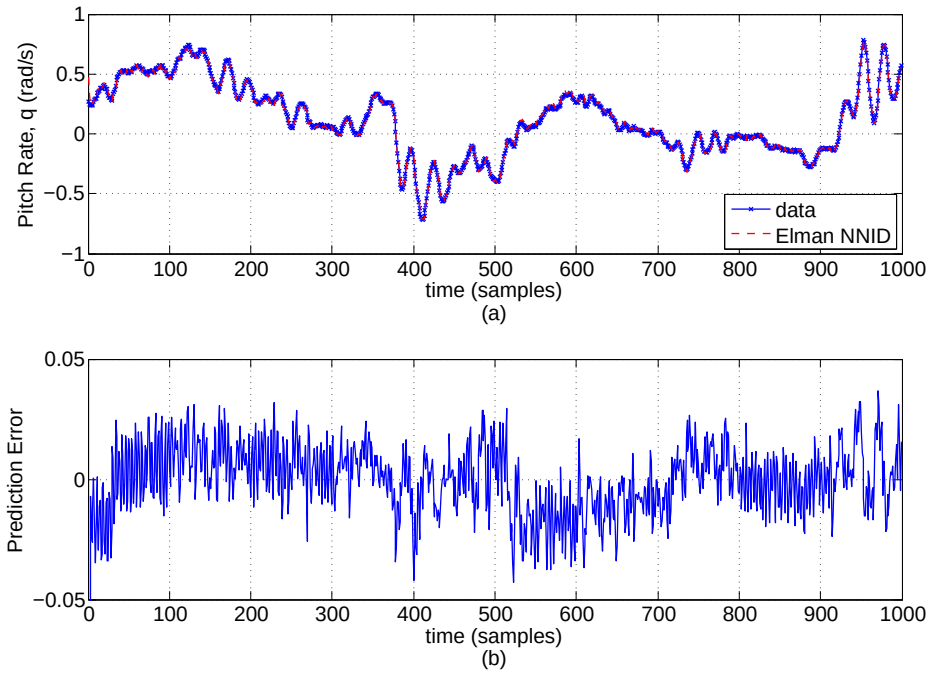


Figure 5.17 The prediction from the modified Elman network for pitch dynamics. (a) The one-step ahead prediction overlaid with the measured helicopter responses. (b) The error plot between one-step ahead prediction and the measurement data. The red dashed line indicates estimation from the modified Elman network while solid blue line with 'x' marker represents the output measurement.

various noise levels indicate that an exceptional prediction performance is achieved up until random noise with $s = 0.5$ added to the optimal weights.

Using the 300 set of weights generated, the 95% confidence intervals can be constructed for the optimal weights using the standard statistical inference method. The average range of the upper and lower output performance (NMCIW) for each noise level case is given in Table 5.9. As can be seen from the result, the weights set added with $s = 0.7$ random noise provides a wide average confidence interval (22.4513%) compare

Table 5.8 The Summaries of Error Statistics for Modified Elman Network.

Error Statistics			
System Responses	RMSE	RMSE (%)	R^2
One-Step Ahead Prediction			
p	0.0604	14.3606	0.9792
q	0.0184	5.7498	0.9962
5-Step Ahead Prediction			
p	0.0967	23.0956	0.9467
q	0.0468	15.6510	0.9755

to the range of measurement between -1 rad/s to 1 rad/s. This indicates the imprecise and high level of uncertainty to produce prediction that represent the real output values.

Table 5.9 The average RMSE for various noise levels applied to optimum weights of the modified Elman network (4 hidden neurons).

Test Error Statistics					
Standard Deviation of Noise	System Responses	RMSE	RMSE (%)	R2	NMCIW (%)
0.01	p	0.0658	15.6337	0.9753	0.0876
	q	0.0174	5.4138	0.9966	0.2425
0.05	p	0.0666	15.8336	0.9747	0.5408
	q	0.0163	5.0892	0.9970	1.1632
0.1	p	0.0661	15.7048	0.9751	0.8776
	q	0.0188	5.8519	0.9961	2.5763
0.3	p	0.0703	16.7037	0.9718	2.7539
	q	0.0699	21.8101	0.9453	6.8574
0.5	p	0.0713	16.9354	0.9710	5.9380
	q	0.0745	23.2384	0.9379	13.1676
0.7	p	0.0985	23.4007	0.9447	12.6861
	q	0.0459	14.3050	0.9765	22.4513

5.5 MODEL PERFORMANCE COMPARISON USING OFF-LINE TRAINING

The prediction performance among three NN architectures proposed in this work is compared by collecting the error statistics that have been generated in the Figure 5.6, 5.11 and 5.15. The NN models performance comparison is made based on the optimal network structures obtained from these figures. Results point that the prediction from NNARX architecture using the MLP network produce prediction quality with a total RMSE percentage of 10.11%. The HMLP network offers prediction with slight performance improvement than the MLP (9.82%) while the modified Elman network gives the lowest prediction quality (15.78%).

Results from Table 5.3, 5.6 and 5.9 indicate that the average RMSE and confidence limit values from the weights are able to produce prediction that adequately fit with the test data. From the results, the prediction performance from the optimised structure of the MLP, HMLP and modified Elman networks are shown to be robust from large variation of weights values. Therefore, it can be conclude that NN training from off-line training method such as LM algorithm would produce weights set that can produce quite large latitude of value but still able to produce a satisfactory prediction performance.

5.6 ON-LINE SYSTEM IDENTIFICATION

In this section, the on-line training proposed in Section 4.3.5 is implemented to identify the attitude dynamics of the UAS helicopter model using the MLP network. The suitable regression vector structure and hidden neurons size for recursive NNARX model can be determined using the k-fold cross validation technique previously discussed. Using results from the off-line system identification, network structure with $n_y = 3$ and $n_u = 1$ is used as the basic model structure for comparing the generalisation performance of the recursive Gauss-Newton (rGN) method with off-line training method.

The comparison also utilises the k-fold cross validation method to identify the efficiency of the selected neural network training methods in estimating the attitude dynamics of the helicopter. To compare the generalisation performance of the rGN

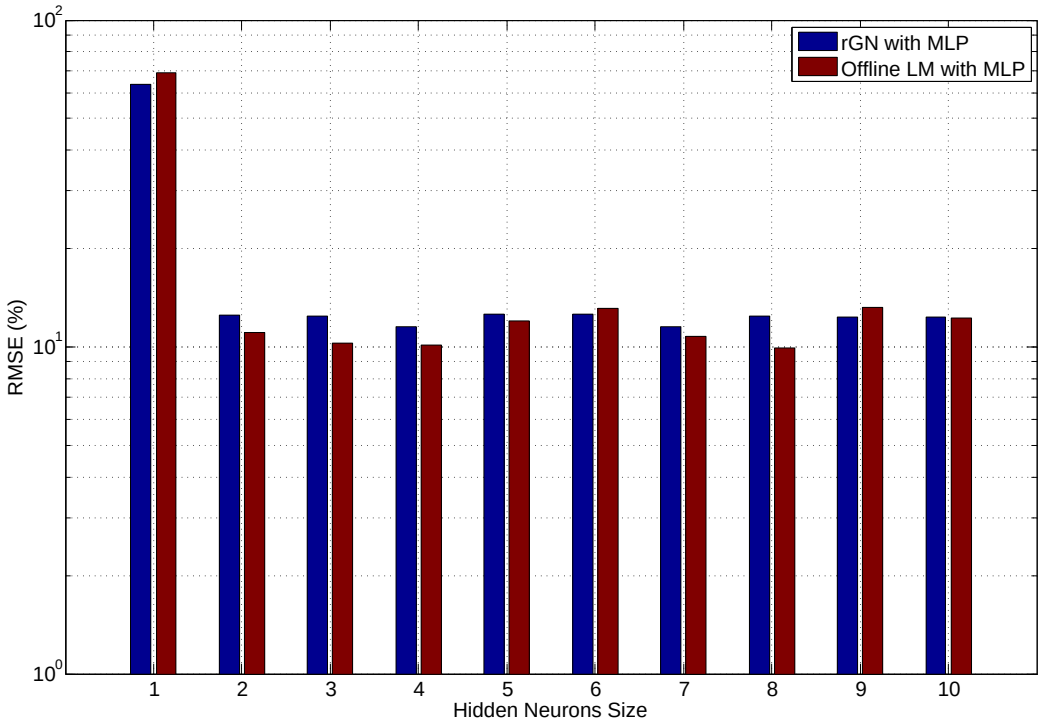
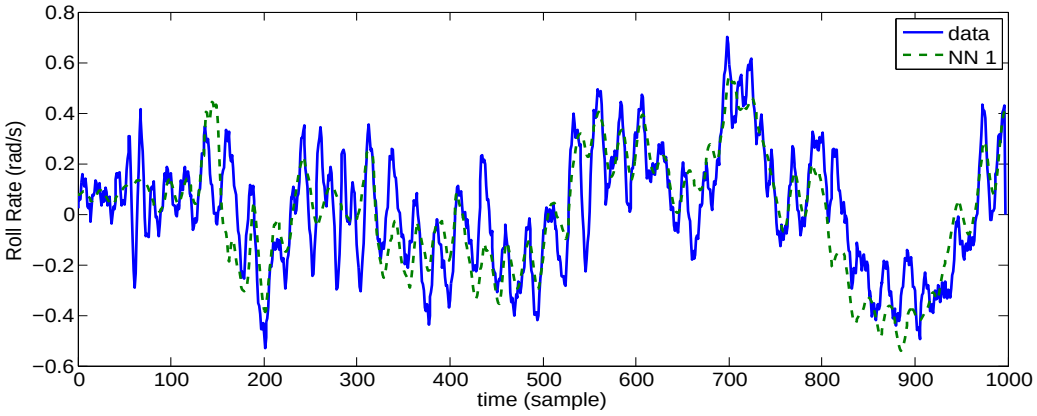


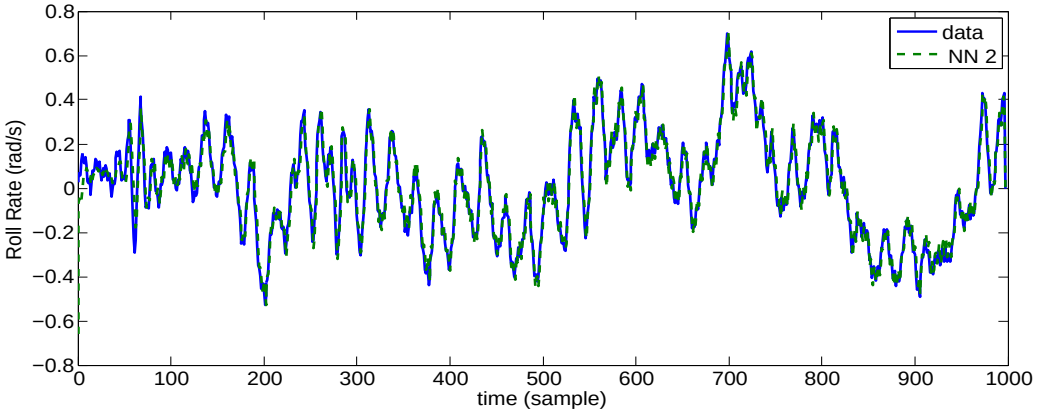
Figure 5.18 The percentage of Root Mean Square Error (RMSE) comparison plot for off-line Levenberg-Marquardt (LM) and recursive Gauss-Newton (rGN) training methods.

method against the off-line LM method, the training of rGN is repeated several times on a finite data set (repeated recursive training) instead of assuming that the data set increases with time as in the recursive training scheme. The general implementation of the repeated recursive training is previously illustrated in Figure 4.10. After reaching the maximum iterations or performance index threshold, the resulting parameter vector θ is then selected for cross validation. The rGN method's training parameters are initialised as $Q(0) = 25I$, $\lambda_0 = 0.99$ and $\lambda(0) = 0.997$. Figure 5.18 indicates the generalisation error plot for repeated rGN and off-line LM methods. The recursive training algorithm (rGN) exhibits a slightly higher generalisation error in cross validation compared with the off-line method. This indicates that training performed over a large data set would give better generalisation performance over the recursive method.

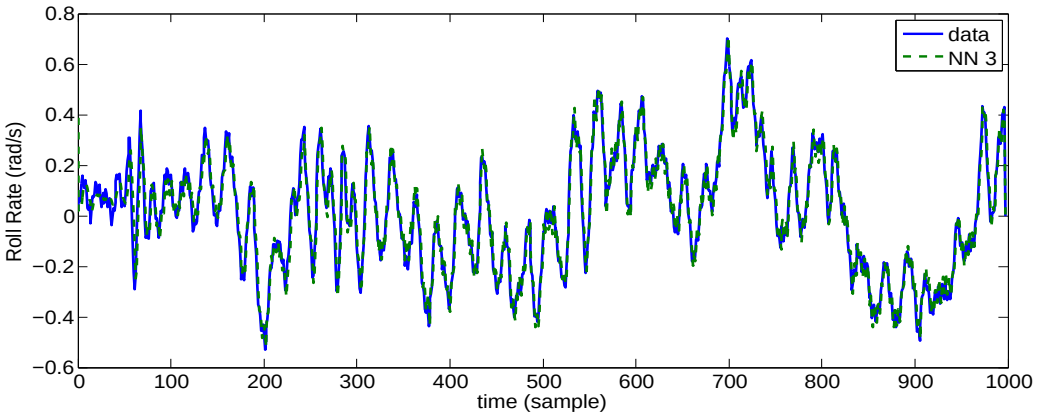
Even though the generalisation error of rGN is slightly higher than off-line LM, the rGN is more adaptive to the changes in dynamic properties. As a comparative study of the adaptability between off-line LM and rGN methods, the roll rate measurement from a new data set is considered. Figure 5.19 shows the prediction from a pre-trained



(a)



(b)



(c)

Figure 5.19 The comparison of the MLP network trained by off-line Levenberg-Marquardt (LM) method and MLP network trained with recursive Gauss-Newton (rGN) method against roll rate measurement. (a) NN model 1 (NN1) is trained with off-line LM algorithm. NN1 model structure is set with $n_y = 1$ and $n_u = 2$ with 4 hidden neurons (b) NN model 2 (NN2) is trained with recursive Gauss-Newton algorithm. NN2 model structure is set with $n_y = 1$ and $n_u = 2$ with 4 hidden neurons; and (c) NN model 3 (NN3) is trained with recursive Gauss-Newton algorithm. NN3 model structure is set with optimised structure from k-fold cross validation ($n_y = 3$ and $n_u = 1$ with 4 hidden neurons).

MLP network using off-line LM (NN1) and MLP network trained by rGN method (NN2 and NN3). The training of rGN model is done once on the roll rate data set. The corresponding error statistic for these prediction models is given in Table 5.10. In Figure 5.19(a) and 5.19(b), the off-line model (NN 1) is pre-trained using 1 past output and 2 past inputs with 4 hidden neurons (training with 1000 samples) while the recursive model (NN 2) is also trained with the same model structure. The training of rGN is carried out using the sliding window method where older data is discarded from the window to allow present data to enter. Thus, it can be seen that the off-line model follows the output measurement accurately at the beginning of the data length and its prediction begins to deteriorate for the remaining data. Whereas, the prediction from the model trained using rGN algorithm adapts well to the dynamic changes that occur during flight even though it was not trained using the optimal model structure. In Figure 5.19(c), the prediction model (NN 3) using the optimal model structure ($n_y = 3$, $n_u = 1$ with 4 hidden neurons) gives the best RMSE accuracy with 19.454% and 11.611% for roll rate and pitch rate respectively. This indicates that improvement in the prediction quality can be achieved if the NN model was trained using optimal model structure. Note that the RMSE values for recursive training (NN3) is slightly higher than results obtained in Figure 5.18 since the recursive training is done with a single pass to the data compared with the results obtained in repeated recursive training.

Table 5.10 Summaries of Error Statistics for model NN1, NN2 and NN3.

Error Statistics				
Training	System Responses	RMSE	RMSE (%)	R2
Off-line LM (NN 1)	p	0.08891	42.376	0.8506
	q	0.03495	16.657	0.9647
rGN (NN 2)	p	0.04907	23.366	0.9451
	q	0.03665	17.454	0.9691
rGN (NN 3)	p	0.09289	19.454	0.9620
	q	0.04244	11.611	0.9863

In an on-line system identification for an adaptive control application, the training time for NN model needs to be less than the sampling time of the control loop. This is

essential since the control decisions need to be updated at the specific timing requirement (22 ms). There are two types of recursive algorithm methods that exist to approximate the non-linear dynamics in real-time; a) mini-batch methods [Samal, 2009, Puttige, 2009], and b) recursive prediction method as presented in previous Section 4.3.5. For mini-batch wise methods, off-line training such as LM algorithm was used to train the NN in real-time by choosing smaller data length to achieve faster convergence time. Typically, a fixed amount of input-output data is collected and stored in a queue. Table 5.11 shows the average training time for mini-batch LM and rGN training algorithms for attitude dynamics identification using the optimal MLP model structure ($n_y = 3$, $n_u = 1$ with 4 hidden neurons). The minimum criterion error (MSE) was selected at 0.001, 0.01 and 0.05 as stopping criteria for mini-batch LM training. The training time comparison test was conducted using a 400 MHz National Instrument's real-time embedded controller. Result from the comparison test shows that mini-batch LM training produces faster training convergence with smaller batch sizes. However, mini-batch LM method requires high computation resources and would not finish within the targeted control sampling period (22 ms). Attempts to reduce the training time of the NN training through manipulation of target MSE values could improve the algorithm training performance, but at the expense of poor training error. A recursive training algorithm such as rGN usually demonstrates faster prediction updates and offers rapid computation of weight adaptation with average training time of 3.88 ms. The average training time for rGN algorithm is well below the control loop sampling period (22 ms) and this indicates that such recursive training algorithms are well suited for real-time applications.

Table 5.11 Training time comparison between mini-batch LM method and rGN method. The values in bracket indicate the total training error (% RMSE).

Average Training Time (ms)						
	Target MSE	Data Sample N=5	N=10	N=15	N=20	N=25
mini batch LM 1	0.001	38.29 [5.02%]	44.98 [6.61%]	46.373 [7.13%]	50.84 [7.53%]	54.78 [7.89%]
mini batch LM 2	0.01	24.48 [13.19]	25.11 [15.72%]	24.89 [16.87%]	26.38 [18.00%]	27.53 [18.47]
mini batch LM 3	0.05	23.22 [22.07%]	23.13 [24.16%]	26.38 [26.83%]	27.37 [28.16]	28.86 [28.48]
		Data Sample N=1				
rGN		3.88 [5.50%]				

5.7 MODEL PERFORMANCE COMPARISON USING RECURSIVE TRAINING

Prediction performance analysis of the MLP, HMLP and Modified Elman networks are repeated in this section using the recursive training algorithm (rGN) to identify whether the proposed network architectures (HMLP and modified Elman network) improve the prediction performance over the standard MLP network. The NN models are trained to predict the future response of helicopter attitude dynamics with 2 output variables.

Since the network architectures of the MLP, HMLP and modified Elman networks are different from each other, the network models should be compared at their best model structure. The model specifications of MLP, HMLP and modified Elman network in Table 5.1, 5.4 and 5.7 are used to carry out the prediction performance comparison analysis in this section. The structure for the MLP network for this analysis is 8 input nodes, 4 hidden neurons (8-4-2) and for the HMLP network, 6 input nodes and 3 hidden neurons are used for prediction (6-3-2). The optimal model structure for the modified Elman network is set with 4 input nodes and 4 hidden neurons (4-4-2). The rGN design parameters are initialised as $Q(0) = 25I$, $\lambda_0 = 0.99$ and $\lambda(0) = 0.997$. Figure 5.20 shows the plot of R^2 values calculated for the MLP, HMLP and modified Elman networks using an independent data set (data length = 10000 samples). The results from the plot

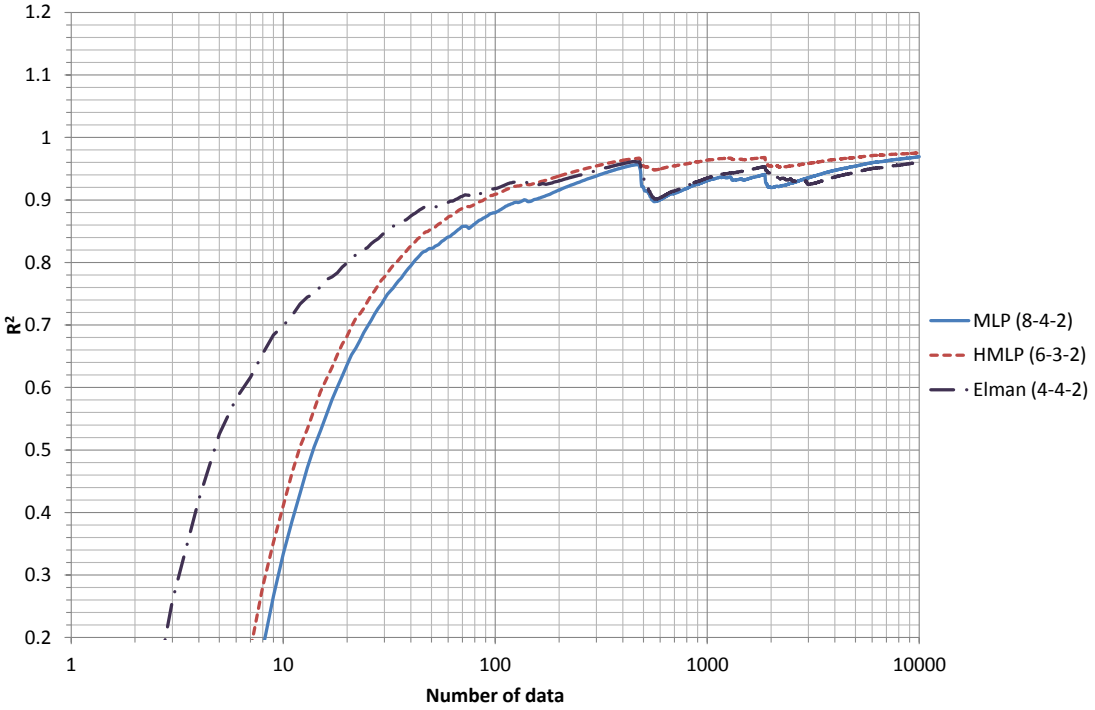


Figure 5.20 The on-line prediction performance comparison for the MLP, HMLP and modified Elman networks. All of these networks are trained by rGN method with optimum model structure identified from off-line model identification.

shows that R^2 values converge to unity over certain sample range. By taking R^2 test value about 0.9, the result shows that the MLP, HMLP and modified Elman networks start to produced good prediction after 155, 92 and 68 test data respectively. All three NN network structures are able to produce good on-line prediction of helicopter attitude dynamics. Nevertheless, the HMLP and modified Elman provide slight improvements to the network's learning rate due to their smaller network structures.

5.8 SUMMARY

In this chapter, the off-line and on-line based system identification methods proposed in Chapter 4 were implemented to identify the attitude dynamics of the helicopter UAS. Three types of NN architectures were used to model the attitude dynamics of the helicopter; namely the MLP, HMLP and modified Elman networks.

NN based modelling technique has a tendency to over-fit the training data since the NN model is consists of large amount of free parameters (network weights) to be determined. The over-fitting problem of NN modelling can be avoided by employing

methods such as weight pruning or introducing regularisation term into the LM training algorithm. For system identification using the off-line LM training, results show training with regularisation term introduces a smoothing effect on the error criterion $V_N(\theta, Z_N)$ in such a way that weights that have less important influence on error are forced to decay towards zero [Samarasinghe, 2007]. In this process, only the important weights that minimise the error are allowed to grow and stabilise at their optimum values. The implementation of regularisation term during NN training prevents the trained NN model from over-fitting that would occur when the NN model is presented with a new test data set.

In order to get a better prediction performance from the NN model, the model structure of the NN model can be further improved through proper network structure selection. To select the NN model structure, the Lipschitz coefficient calculation and k-cross-validation test methods were proposed, and they were used to identify the optimal or near optimal model structure of the NN model without attempting the tedious trial and error approach. The validation result shows that the model structure of the MLP architecture can be identified correctly with 3 past outputs and 1 past input using the proposed methods. Further comparison with model structure selection from previous studies such as in Samal [2009] and Putro et al. [2009] show that the identified model structure using k-cross validation offers an improvement in terms of generalisation error. Similarly, the minimum number of neurons to be included in the NN model can also be selected using the proposed validation methods.

The HMLP and modified Elman networks were proposed in this work to provide us with simpler representation of UAS dynamics and reduction in total number of weight connections used in the NN model. Furthermore, the reduction in the total number of weights in the network can significantly reduce the computation time needed to train the NN model. Similar methodologies to select the optimal structure in the MLP network was used to identify the network structure for both HMLP and modified Elman networks. Based on validation test results, the model structures of the HMLP and modified Elman networks are found to be much smaller than the standard MLP network. Although the total number of weights for the HMLP and modified Elman

networks are lower than the MLP network, the prediction performance of both NN models are on par with the prediction quality of the MLP network. The results also point out that the average RMSE and the confidence limit values from each of the proposed networks are able to produce prediction that adequately fit with the test data. The prediction from the MLP, HMLP and modified Elman networks are also shown to be robust from large variations of weights values. Therefore, it can be concluded that NN training from the off-line training methods such as the LM algorithm or repeated GN method are able to produce satisfactory prediction performance even with large deviation of weights set derived from the training process.

Validation results conducted in this study confirmed that the off-line NN model is suitable for modelling the helicopter's attitude dynamics correctly. However, the dynamic model identified using the off-line NN model has a major drawback such that the method's inability to represent the entire flight operation very well because of the time varying nature of helicopter flight dynamics. Recursive type training such as the rGN method was used in this study to overcome such problems in system identification. Results indicate that the rGN algorithm is more adaptive to the changes in dynamic properties, although the generalisation error of repeated rGN is slightly higher than off-line LM method. The rGN method is also found capable of producing a satisfactory prediction quality even though the model structure was incorrectly selected. The generalisation and adaptability performance of the model can be further improved by properly selecting the optimised network structure with the aid of k-fold cross validation method.

The recursive method presented in this work is suitable for modelling the helicopter in real-time within the control sampling time and computational resource constraints. Recursive Gauss-Newton method used in the NN training demonstrates faster prediction updates and offers rapid computation of weight adaptation with average training time of 3.88 ms. The average training time for rGN algorithm is well below the targeted control loop sampling period (22 ms). This indicates that such recursive training algorithms are well suited for real-time applications. Furthermore, the proposed HMLP and modified Elman networks are found to improve the learning rate of NN prediction and this

would enable the implementation of the real time recursive computation of the NN based system identification models. These models can be further used for the design of adaptive flight controllers for autonomous flight. In the next chapter, the theoretical foundation of the NN based flight controller design is presented.

Chapter 6

NEURAL NETWORK BASED PREDICTIVE CONTROL SYSTEM

6.1 INTRODUCTION

The fundamental concepts of NN based model predictive control (MPC) are presented in this chapter. For the design of an adaptive flight controller for the helicopter UAS, the non-linear models identified from the NN based system identification are employed. The NN based MPC is a NN based control algorithm classified under indirect NN control system design [Agarwal, 1997, Norgaard, 2000]. The design of this type of controller always depends on the availability of the dynamic model of the system. However, the controller design using this approach often follows the conventional controller design while the NN models obtained in advanced are merely used as an aid to the controller development. Some examples of conventional control methods used in indirect NN control designs include approximate pole placement, minimum variance, predictive control, and non-linear predictive control design have been introduced and discussed in detailed in Norgaard [2000].

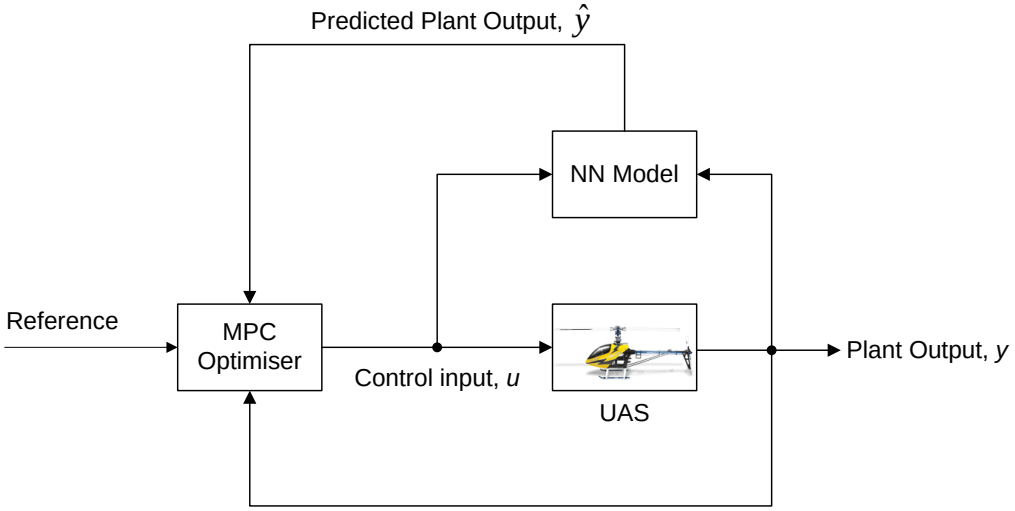
In this chapter, the Neural Network based Approximate Predictive Controller (NNAPC) is discussed. It is designed and developed using the NN model identified from off-line or on-line system identification algorithms. The rest of the chapter is organised as follows. Section 6.2 describes the motivation to use such an approach and the basics of NNAPC design. Section 6.3 provides the details of the instantaneous linearisation concept to extract the necessary linear model for the NNAPC's prediction process. The formulation of the state space model from the linear ARX transfer function model and

addition of an integrator term into the state space model are presented in Section 6.4 and 6.5. Section 6.6 and 6.7 describe the model prediction process using the state space model and optimisation routine for NNAPC design. In Section 6.8, the NNAPC design with constraints is discussed in detail. The operational constraints in the optimisation step are introduced into NNAPC design to improve the performance of the control system. Then, the chapter presents the basic control architectures of helicopter UAS and NNAPC algorithm implementation in Section 6.9. The chapter is summarised in Section 6.10.

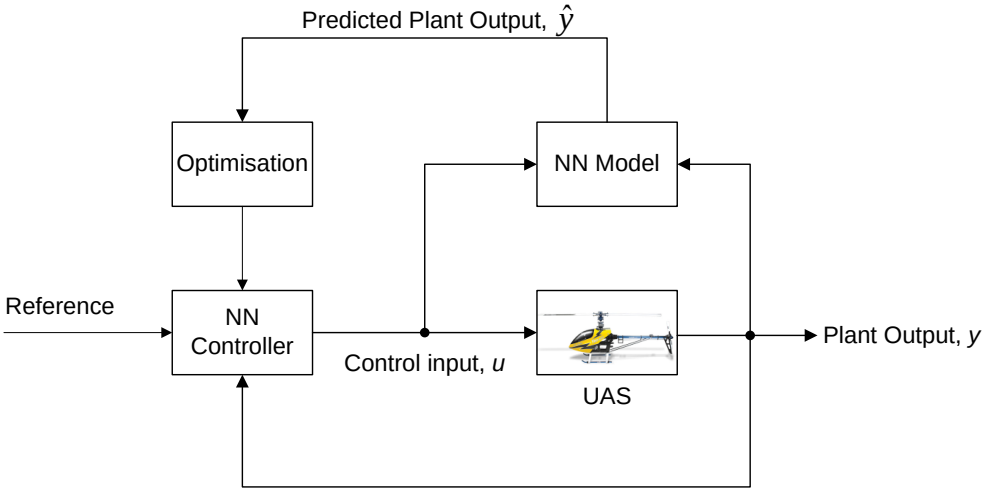
6.2 NN BASED APPROXIMATE PREDICTIVE CONTROL PRINCIPLES

The main objective of the MPC implementation is to compute a trajectory of future manipulated variable u to optimise the future plant output $\hat{y}$ over a specified time horizon [Camacho and Bordons, 2004, Rawlings, 2000]. Figure 6.1(a) shows the basic architecture of the NN based MPC which uses prediction from a NN model to generate future plant output. In this approach, a process model such as NN provides a prediction of the future plant response over the specified horizon before being fed into an optimisation process. The optimisation process is carried out within a limited time horizon by giving the initial plant dynamics at the start of the process. System operational constraints can also be incorporated into the optimisation process which improves the control system performance when the control signals or system states violate the operation constraints. The optimisation process will be responsible for finding the value of the manipulated variable u which minimises a specified performance criterion given the current state measurement and future reference trajectory.

The optimisation procedure using multi-step ahead prediction from the non-linear NN model often results in demanding computation efforts [Witt et al., 2007]. This is due to the minimisation of non-convex optimisation cost criterion because of the introduction of a non-linear model such as NN model [Samal, 2009]. The optimisation problem becomes a complex non-linear programming problem with no guarantee of reaching global minimum as the prediction from the model is determined by the non-linear



(a)



(b)

Figure 6.1 Different configuration of NN based Model Predictive Control (MPC): (a) Basic configurations of NN based MPC which used prediction from a NN model (b) NN based controller that mimics the MPC controller by learning the controller input selection by optimisation process

relationship of the future inputs [Norgaard, 2000]. To derive the control law for MPC, a pure gradient based method as used in Section 4.3.4 for NN training can be used for the optimisation problem. However, the gradient based method in general has problems in finding the minimum solution in real-time. This is due to the complexity of the Gradient and Hessian matrix computations. The computation loading of the MPC is made even worse if a significant number of constraints are imposed on the solution of optimisation problem.

As an alternative to overcome this problem, a NN controller can be trained in off-line mode to mimic a conventional MPC controller. Subsequently this would reduce the computation and the tuning effort. An alternative configuration of NN based MPC is given in 6.1(b). In this configuration, the neural network controller is used to learn the controller input selection process calculated by the MPC optimisation algorithm. The training process of the NN controller is normally conducted off-line and at the end of the training process, the MPC optimisation step is replaced completely by the trained NN controller [Agarwal, 1997, Pottmann and Seborg, 1997]. However, this approach is impractical to implement as the method requires an MPC controller to be present before the implementation of the NN controller. Moreover, the NN controller performance could be compromised due to the nature of the off-line training, where certain segments of the flight operating range could not be properly represented. Furthermore, this approach is also unattractive to pursue since the NN controller needs to be retrained all over again whenever the configuration of MPC controller is modified.

This chapter proposes a solution to the aforementioned drawbacks of NN based MPC real-time implementation by introducing the application of Neural Network based Approximate Predictive Control (NNAPC) using linear model prediction from identified NN model obtained either from off-line or on-line modelling techniques. Figure 6.2 shows the block diagram of NNAPC architecture. The NNAPC uses the linearised model extracted from the NN model through the principle of instantaneous linearisation [Norgaard, 2000]. Subsequently, this would make the NNAPC to be less computational and less demanding compared with other MPC techniques such as NN based MPC or non-linear model predictive control (NMPC) [Ogunfunmi, 2007]. Furthermore, the NNAPC design parameters are easy to tune and are found effective for a wide range of control applications and are suitable for systems with time delay [Norgaard, 2000, Witt et al., 2007, Lawrynczuk, 2007b]. The main difference between NNAPC with linear MPC and other NMPC methods lies in the prediction operation of NNAPC. Instead of relying on the prediction from single linear or non-linear model, the NNAPC controller extracts a linear model from the identified NN model at each sampling time and uses it to predict the future plant response within the specified time horizon.

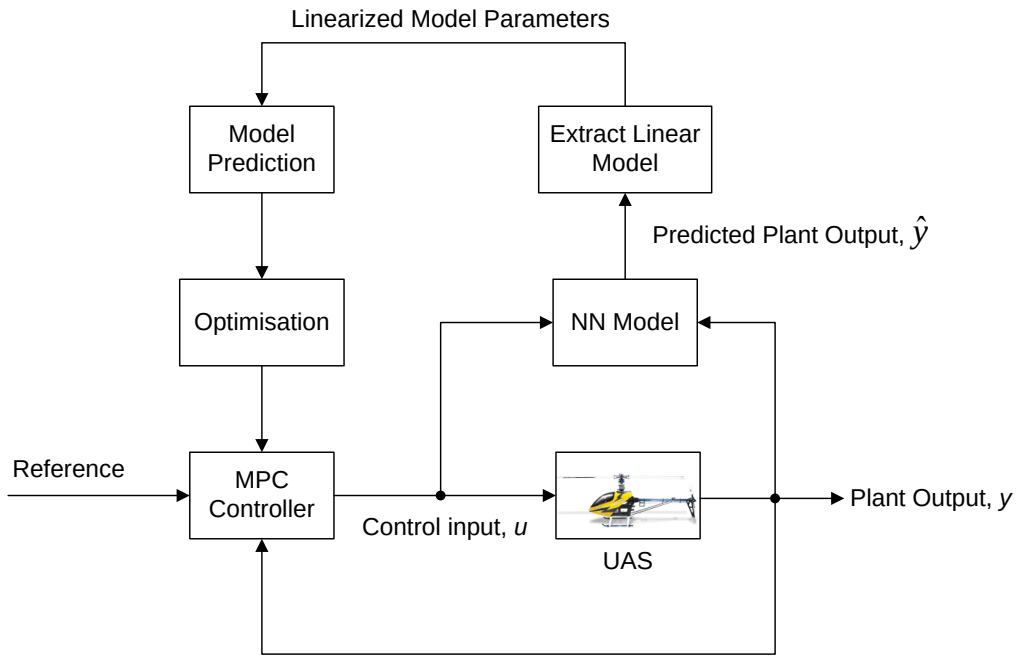


Figure 6.2 The Neural Network based Approximate Predictive Control (NNAPC) scheme based on instantaneous linearisation of the NN model.

As mentioned before, the objective of NNAPC controller is to bring the predicted output of the system $\hat{y}$ as close as possible to a specified set-point $r(k)$ at sample time step k. Here, the specified set-point is assumed to be constant in a single optimisation window. The best control parameter vector ΔU that reduces the error difference between set-point and predicted output over the N_c control horizon is found by minimising the objective function $J(k)$. The objective function $J(k)$ that reflects the NNAPC objective is given as follows:

$$J(k) = \left(R_s - \hat{Y}\right)^T \left(R_s - \hat{Y}\right) + \Delta U^T \bar{R} \Delta U \tag{6.1}$$

where $\hat{Y}$ denotes the future predicted output variables over prediction horizon N_p and ΔU represents the future control trajectory over control horizon N_c. In the case of SISO control, the dimension of the predicted output vector $\hat{Y}$ is $N_p \times 1$ and the dimension of the control trajectory ΔU is $N_c \times 1$. Assuming that the dynamic system has m inputs, n_1 states and q outputs, the predicted output vector $\hat{Y}$ and control trajectory ΔU are still represented in vector form for MIMO case with dimension qN_p and mN_c

respectively. The predicted output vector $\hat{Y}$ and control trajectory ΔU for the MIMO case are defined as:

$$\begin{aligned}\hat{Y} &= [y_1(k+1|k) \quad y_1(k+2|k) \quad y_2(k+1|k) \quad y_2(k+2|k) \quad \cdots \quad y_q(k+N_p|k)]^T \\ \Delta U &= [\Delta u_1(k) \quad \Delta u_1(k+1) \quad \Delta u_2(k) \quad \Delta u_2(k+1) \quad \cdots \quad \Delta u_m(k+N_c)]^T \end{aligned} \tag{6.2}$$

The data vector R_s that contains qN_p set-points is defined by:

$$R_s = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & 1 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}^T}_{qN_p} \begin{bmatrix} r_1(k) \\ r_2(k) \\ \vdots \\ r_q(k) \end{bmatrix} = \bar{R}_s r(k) \tag{6.3}$$

The first term of the objective function in Equation (6.1) is related to the minimisation of error between predicted output variables and predefined set-point. Subsequently, the second term in the objective function indicates the treatment of ΔU when minimising the objective function $J(k)$. In Equation (6.1), the variable matrix R represents a diagonal matrix in the form of $R = r_w I_{mN_c \times mN_c} \quad (r_w \geq 0)$. The constant r_w is used as a tuning parameter for the desired closed-loop performance. The closer r_w value to zero would cause the optimisation to prioritise the minimisation of error between the predicted output variables and the predefined set-points to the smallest value possible, without considering the magnitude and the smoothness of the control trajectory ΔU.

In general, the implementation principle of NNAPC control shown in Figure 6.2 can be summarised as follows:

1. The internal dynamic model of the system is used to predict the future output response of the system. This model can be obtained and implemented either from off-line or on-line system identification algorithms.

2. Using instantaneous linearisation principle, a linear model is extracted from the NNARX model and converted to corresponding state space model. This state

space model obtained at every sampling instance is used to predict the future output response $\hat{Y}$ over a specified prediction horizon N_p.

3. A suitable reference trajectory R_s is obtained from the specified output trajectory $r(k)$ over a specified prediction horizon N_p.

4. The cost function $J(k)$ is constructed using the predicted response $\hat{Y}$ and the reference trajectory, R_s.

5. A set of future control trajectory ΔU is calculated by minimising the cost function $J(k)$ to achieve the desired tracking response.

6. Apply the first element of the calculated future control trajectory ΔU as the actual control input to drive the system under consideration.

7. The procedure is repeated to calculate a new output prediction and future control trajectory using the sensor measurement at the next sample time.

6.3 PRINCIPLE OF INSTANTANEOUS LINEARISATION

In order to implement the NNAPC scheme, the linearised NNARX model is extracted from the non-linear NN model (MLP, HMLP or Elman networks) at every time sample k. Given that a non-linear NN model of the dynamic system is:

$$\hat{y}(k) = g\left[\varphi(k)\right] \tag{6.4}$$

where the regression vector is given as follows:

$$\varphi(k) = \left[y(k-1) \quad \cdots \quad y(k-n_y) \qquad u(t-k) \quad \cdots \quad u(k-n_u)\right]^T \tag{6.5}$$

The approximate linear model is obtained by linearising $g(\varphi(k))$ around the current state $\varphi(\tau)$. The linear model is given as follows:

$$\begin{aligned}\widetilde{y}(k) = &-a_1\widetilde{y}(k-1) - a_2\widetilde{y}(k-2)\cdots - a_{n_y}\widetilde{y}(k-n_y) \\ &+ b_0\widetilde{u}(k) + b_1\widetilde{u}(k-1) + \cdots + b_{n_u}\widetilde{u}(k-n_u)\end{aligned} \tag{6.6}$$

where,

$$a_{n_y} = -\frac{\partial \hat{y}(k)}{\partial y(k-n_y)}\bigg|_{\varphi(k)=\varphi(\tau)}; \quad b_{n_u} = -\frac{\partial \hat{y}(k)}{\partial u(k-n_u)}\bigg|_{\varphi(k)=\varphi(\tau)} \tag{6.7}$$

and,

$$\begin{aligned} \widetilde{y}(k-n_y) &= y(k-n_y) - y(\tau-n_y); \\ \widetilde{u}(k-n_u) &= u(k-n_u) - u(\tau-n_u) \end{aligned} \tag{6.8}$$

The constant n_y and n_u are the size of the past output and input measurements. The approximate model in Equation (6.6) can be alternatively expressed similar to Equation (4.18), where the coefficients a_{n_y} and b_{n_u} are collected in the polynomial $A(q^{-1})$ and $B(q^{-1})$ as follows:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + \cdots + a_{n_y} q^{-n_y}; \\ B(q^{-1}) &= b_0 + b_1 q^{-1} + \cdots + b_{n_u} q^{-n_u} \end{aligned} \tag{6.9}$$

The instantaneous linearisation of the NN model can be derived by taking the partial derivative of the NN model prediction with respect to each system input [Norgaard, 2000, Lawrynczuk, 2007b,a]. Applying the chain rules to Equation (4.36) with respect to regression vector, the linearisation model for the MLP network with tangent hyperbolic and linear activation function in hidden and output units yields:

$$\begin{aligned} \frac{\partial \hat{y}_i(k)}{\partial \varphi_j(k)} &= \sum_{h=1}^{H} W2_{ih} W1_{hj} \left[1 - tanh^2 \left(\sum_{j=1}^{m} W1_{hj} \varphi_j(t) + B1_h \right) \right] \\ &\text{with} \quad h = 1, 2, 3 \cdots H \quad \text{and} \quad i = 1, 2, 3 \cdots n \end{aligned} \tag{6.10}$$

For the case of linear units in both hidden and output layers, the linearisation is given by:

$$\begin{aligned} \frac{\partial \hat{y}_i(k)}{\partial \varphi_j(k)} &= \sum_{h=1}^{H} W2_{ih} W1_{hj} \\ &\text{with} \quad h = 1, 2, 3 \cdots H \quad \text{and} \quad i = 1, 2, 3 \cdots n \end{aligned} \tag{6.11}$$

Similarly, the linear model extracted from the linearisation of HMLP network with tangent hyperbolic and linear activation function in hidden and output units is given by:

$$\frac{\partial \hat{y}_i(k)}{\partial \varphi_j(k)} = \sum_{h=1}^{H} W2_{ih} W1_{hj} \left[1 - tanh^2 \left(\sum_{j=1}^{m} W1_{hj} \varphi_j(t) + B1_h \right) \right] + \sum_{j=1}^{m} W3_{ij}$$
$$\text{with} \quad h = 1, 2, 3 \cdots H \quad \text{and} \quad i = 1, 2, 3 \cdots n \tag{6.12}$$

For the HMLP network with linear activation function in both hidden and output units, the linearisation term is expressed as:

$$\frac{\partial \hat{y}_i(k)}{\partial \varphi_j(k)} = \sum_{h=1}^{H} W2_{ih} W1_{hj} + \sum_{j=1}^{m} W3_{ij}$$
$$\text{with} \quad h = 1, 2, 3 \cdots H \quad \text{and} \quad i = 1, 2, 3 \cdots n \tag{6.13}$$

Finally, the linear model extracted from the modified Elman network with tangent hyperbolic and linear activation functions for hidden and output processing units is formulated such as:

$$\frac{\partial \hat{y}_i(k)}{\partial \varphi_j(k)} = \sum_{h=1}^{H} W2_{ih} W1_{hj} \left[1 - tanh^2 \left(\sum_{j=1}^{m} W1_{hj} \varphi_j(t) + B1_h + \sum_{k=1}^{m} W3_k x_k(k) \right) \right]$$
$$\text{with} \quad h = 1, 2, 3 \cdots H \quad \text{and} \quad i = 1, 2, 3 \cdots n \tag{6.14}$$

For linear activation function in hidden and output layers, the linear model for Elman network is given as:

$$\frac{\partial \hat{y}_i(k)}{\partial \varphi_j(k)} = \sum_{h=1}^{H} W2_{ih} W1_{hj}$$
$$\text{with} \quad h = 1, 2, 3 \cdots H \quad \text{and} \quad i = 1, 2, 3 \cdots n \tag{6.15}$$

6.4 NON-MINIMAL STATE SPACE MODEL REALISATION

There are generally three types of models used in MPC design such as finite impulse response (FIR)/step response models, transfer function models and state space models [Rossiter, 2003]. The state space model is preferred over other model types in MPC controller design due to its simplicity and effective handling of multi-variable problems. The transfer function models obtained from instantaneous linearisation of the NN model can also be represented in terms of state space model realisation. This can be done by reformulating the state variables of the state space model to be identical to the feedback variables that have been used in the ARX model [Wang, 2009b, Ordys and Clarke, 1993]. Consider the general discrete time ARX model that describes the input and output relationship as:

$$A(q^{-1})(1-q^{-1})y(k) = B(q^{-1})\Delta u(k) + q^{-1}\epsilon(t) \tag{6.16}$$

where $\epsilon(t)$ is the input disturbance which is assumed to be a sequence of integrated white noise, $A(q^{-1})$ and $B(q^{-1})$ are polynomials in the time shift operator given by the following forms:

$$\begin{aligned} A(q^{-1}) &= 1 + a_1 q^{-1} + a_2 q^{-2} \cdots + a_n q^{-n} \\ B(q^{-1}) &= b_1 q^{-1} + b_2 q^{-2} \cdots + b_n q^{-n} \end{aligned} \tag{6.17}$$

Let the polynomial $A(q^{-1})(1-q^{-1})$ be referred as:

$$A(q^{-1})(1-q^{-1}) = 1 + \bar{a}_1 q^{-1} + \bar{a}_2 q^{-2} + \cdots + \bar{a}_{n+1} q^{-(n+1)} \tag{6.18}$$

Next, by choosing the state variable vector as:

$$x(k) = [y(k)\ y(k-1)\ \cdots\ y(k-n-1)\ \Delta u(k-1)\ \Delta u(k-2)\ \cdots\ \Delta u(k-n)]^T \tag{6.19}$$

the corresponding state space model with non-minimal realisation is formulated as:

$$x_m(k+1) = A_m x_m(k) + B_m \Delta u(k) + B_d \epsilon(k) \tag{6.20}$$

$$y(k) = C_m x_m(k) \tag{6.21}$$

where A_m denotes a square matrix of $(2n+1) \times (2n+1)$ size, B_m is a matrix of size $(2n+1) \times 1$ and C_m is a matrix of $1 \times (2n+1)$ size. The detailed formulation of matrix A_m, B_m and C_m are given as follows:

$$A_m = \begin{bmatrix} -\bar{a}_1 & -\bar{a}_2 & \cdots & -\bar{a}_{n-1} & -\bar{a}_n & b_2 & \cdots & b_{n-1} & b_n \\ 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}; \quad B_m = \begin{bmatrix} b_1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$C_m = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \end{bmatrix}; \quad B_d = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

6.5 GENERAL FORMULATION OF AUGMENTED MODEL

The design of predictive control is based on the dynamic model of the system. The state space model approach is used in this work to design the predictive controller for the system under consideration. By using a state-space model representation, the current state variable information is used to predict the future response of the system. The MPC controller design used in this work is adapted from Wang [2009a], which requires an integrator term to be embedded into the state space model of the plant. The general formulation of the model is described in the following.

Assuming a MIMO plant has m inputs, n_1 states and q outputs, the basic formulation of the state space model is given in Equation (6.20). By taking difference operation on both sides of Equation (6.20), we obtain:

$$\begin{aligned} x_m(k+1) - x_m(k) = A_m \left(x_m(k) - x_m(k-1)\right) + B_m \left(u(k) - u(k-1)\right) \\ + B_d \left(\epsilon(k) - \epsilon(k-1)\right) \end{aligned} \tag{6.22}$$

Introducing $\Delta x_m(k+1) = x_m(k+1) - x_m(k)$, $\Delta x(k) = x_m(k) - x_m(k-1)$, $\Delta u(k) = u(k) - u(k-1)$ and $w(k) = \epsilon(k) - \epsilon(k-1)$, the difference of the state-space equation is:

$$\Delta x_m(k+1) = A_m \Delta x_m(k) + B_m \Delta u_m(k) + B_d w(k) \tag{6.23}$$

To relate the state variable $\Delta x_m(k)$ with the output $y(k)$, the following formulation is introduced:

$$\begin{aligned} \Delta y(k+1) &= C_m \Delta x_m(k+1) \\ &= C_m A_m \Delta x_m(k) + C_m B_m \Delta u(k) + C_m B_d w(k) \end{aligned} \tag{6.24}$$

where $\Delta y(k+1) = y(k+1) - y(k)$.

If $x(k) = \begin{bmatrix} \Delta x_m(k)^T & y(k)^T \end{bmatrix}^T$ is introduced as a new state variable vector, the new

state space model equation is given:

$$\begin{bmatrix} \Delta x_m(k+1) \\ y(k+1) \end{bmatrix} = \begin{bmatrix} A_m & O_m^T \\ C_m A_m & I_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} + \begin{bmatrix} B_m \\ C_m B_m \end{bmatrix} \Delta u(k) + \begin{bmatrix} B_d \\ C_m B_d \end{bmatrix} \Delta w(k)$$
$$y(k) = \begin{bmatrix} O_m & I_{q\times q} \end{bmatrix} \begin{bmatrix} \Delta x_m(k) \\ y(k) \end{bmatrix} \tag{6.25}$$

where O_m is a $q \times n_1$ zero matrix and $I_{q\times q}$ is an identity matrix with $q \times q$ dimensions. Then, for notation simplicity, Equation (6.25) is denoted as:

$$x(k+1) = Ax(k) + B\Delta u(k) + B_d w(k)$$
$$y(k) = Cx(k) \tag{6.26}$$

Since the original state space model is augmented with integrators and MPC controller is based on the augmented model, it is important to ensure that the augmented system plant meets controllable and observable conditions, especially with respect to unstable system [Wang, 2009c]. Controllability is a pre-requisite for the predictive control system to achieve the desired closed-loop control performance, and observability is a pre-requisite for the successful design of an observer.

6.6 PREDICTION FROM THE STATE SPACE MODELS

After formulation of the state space model, the next step in MPC control design is to calculate the predicted plant output variables with the future control signal as the adjustable variable. In this section, the prediction from the state space model is described within a single optimisation window with N_p sample length. Assume that at the current sampling time k, the state variable vector $x(k)$ is available through measurement. The future state variables predicted from the state space model over N_p sample length can be represented in a sequence as:

$$x(k+1 \mid k),\ x(k+2 \mid k),\ \cdots,\ x(k+N_p \mid k)$$

Similarly, the sequence of future control variables over the control horizon N_u are denoted by:

$$\Delta u(k),\ \Delta u(k+1),\ \cdots,\ \Delta u(k+N_c-1)$$

where the control horizon N_c is chosen to be equal or less to the prediction horizon N_p. The future state variables can be determined sequentially using the set of the future control variables. Based on matrix A, B and C from Equation (6.26), the future state variables are updated as follows:

$$\begin{aligned}
x(k+1 \mid k) &= Ax(k) + B\Delta u(k) + B_d w(k) \\
x(k+2 \mid k) &= Ax(k+1 \mid k) + B\Delta u(k+1) + B_d w(k+1 \mid k) \\
&= A^2 x(k) + AB\Delta u(k) + B\Delta u(k+1) \\
&\quad + AB_d w(k) + B_d w(k+1 \mid k) \\
&\vdots \\
x(k+N_p \mid k) &= A^{N_p} x(k) + A^{N_p-1} B\Delta u(k) + A^{N_p-2} B\Delta u(k+1) \\
&\quad + A^{N_p-N_c} B\Delta u(k+N_c-1) + A^{N_p-1} B_d w(k) \\
&\quad + A^{N_p-2} B_d w(k+1 \mid k) + \cdots + B_d w(k+N_p-1 \mid k)
\end{aligned} \tag{6.27}$$

Similarly, the predicted output variables are derived from the predicted state variables by substitution leading to:

$$\begin{aligned}
y(k+1 \mid k) &= CAx(k) + CB\Delta u(k) + CB_d w(k) \\
y(k+2 \mid k) &= CAx(k+1 \mid k) + CB\Delta u(k+1) + CB_d w(k+1 \mid k) \\
&= CA^2 x(k) + CAB\Delta u(k) + CB\Delta u(k+1) \\
&\quad + CAB_d w(k) + CB_d w(k+1 \mid k) \\
&\vdots \\
y(k+N_p \mid k) &= CA^{N_p} x(k) + CA^{N_p-1} B\Delta u(k) + CA^{N_p-2} B\Delta u(k+1) \\
&\quad + CA^{N_p-N_c} B\Delta u(k+N_c-1) + CA^{N_p-1} B_d w(k) \\
&\quad + CA^{N_p-2} B_d w(k+1 \mid k) + \cdots + CB_d w(k+N_p-1 \mid k)
\end{aligned} \tag{6.28}$$

Here, the $w(k)$ is a zero mean white noise sequence and the future predicted value of $w(k+1 \mid k)$ is assumed to be zero. Note that the predicted output variables (6.28) and the predicted state variable (6.27) are presented using only the current state variable information $x(k)$ and sequence of current and future control movements $\Delta u(k+j)$, where $j = 0, 1, 2, \cdots, N_c - 1$. For notation simplicity, the expectation operator can be omitted. By defining the following vectors, (similar to definition in Equation (6.2)):

$$\begin{aligned}
\hat{Y} &= [y(k+1 \mid k) \quad y(k+2 \mid k) \quad y(k+3 \mid k) \quad \cdots \quad y(k+N_p \mid k)]^T \\
\Delta U &= [\Delta u(k) \quad \Delta u(k+1) \quad \Delta u(k+2) \quad \cdots \quad \Delta u(k+N_c-1)]^T
\end{aligned}$$

the formulation for the predicted output variables (6.28) can be represented in a compact matrix form as:

$$\hat{Y} = \Gamma x(k) + \Phi \Delta U \tag{6.29}$$

where,

$$\Gamma = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ \vdots \\ CA^{N_p} \end{bmatrix}; \quad \Phi = \begin{bmatrix} CB & 0 & 0 & \cdots & 0 \\ CAB & CB & 0 & \cdots & 0 \\ CA^2B & CAB & CB & \cdots & 0 \\ \vdots & & & & \\ CA^{N_p-1}B & CA^{N_p-2}B & CA^{N_p-3}B & \cdots & CA^{N_p-N_c}B \end{bmatrix} \tag{6.30}$$

Again, for the SISO case, the dimension of vector $\hat{Y}$ is $N_p \times 1$ and the dimension of vector ΔU is $N_c \times 1$. The dimension of matrix Γ is $N_p \times n$ and matrix Φ is $N_p \times N_c$. Considering a MIMO case, the dimension for $\hat{Y}$ and ΔU are vectors $mN_p \times 1$ and $qN_c \times 1$. Subsequently, the dimension of matrix Γ and Φ for MIMO case is set to $qN_p \times n$ and $qN_p \times mN_c$ respectively.

6.7 MODEL PREDICTIVE CONTROL OPTIMISATION

The main objective of predictive control design is to bring the predicted output of the system as close as possible to the reference signal within the prediction horizon, for a given reference signal $r(k)$ at sample time k. Here, the reference signal $r(k)$ is assumed to remain constant throughout the optimisation window. This could be achieved by finding the optimal solution of control parameter vector ΔU such that the error cost function between the reference signal $r(k)$ and predicted output $\hat{Y}$ is minimised.

In order to find the optimal solution of ΔU that minimises Equation (6.1), the cost function is expressed as the following form after substituting Equation (6.29) into (6.1):

$$\begin{aligned} J = (R_s - \Gamma x(k))^T (R_s - \Gamma x(k)) - 2\Delta U^T \Phi^T (R_s - \Gamma x(k)) \\ + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U \end{aligned} \tag{6.31}$$

Taking the first derivative of the cost function J:

$$\frac{\partial J}{\partial \Delta U} = -2\Phi^T (R_s - \Gamma x(k)) + 2(\Phi^T \Phi + \bar{R}) \Delta U \tag{6.32}$$

the necessary condition to obtain the minimisation of J is by setting:

$$\frac{\partial J}{\partial \Delta U} = 0$$

which leads to the following solution for the incremental control signal within one optimisation window:

$$\Delta U = (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - \Gamma x(k)) \tag{6.33}$$

where the matrix $\Phi^T \Phi$ has dimension of $mN_c \times mN_c$, matrix $\Phi^T \Gamma$ has dimension $mN_c \times n$ and matrix $\Phi^T \bar{R}_s$ has dimension $mN_c \times q$. Matrix $\bar{R}$ denotes a diagonal matrix with dimension equal to $\Phi^T \Phi$. The matrix $\bar{R}$ takes the form of $\bar{R} = r_w I_{mN_c \times mN_c}$ $(r_w \geq 0)$ with r_w used as control penalty factor to achieve the desired closed loop performance. The reference signal is given by $r(k) = [r_1(k) \quad r_2(k) \quad r_3(k) \cdots r_q(k)]^T$, which indicates the q reference signals to the multi output system.

Since the MPC control only uses the first m elements in ΔU, the final form of incremental optimal control is given as follows:

$$\begin{aligned} \Delta U &= \underbrace{[I_m \quad O_m \cdots O_m]}_{N_c} (\Phi^T \Phi + \bar{R})^{-1} \Phi^T (R_s - \Gamma x(k)) \\ &= K_y r(k) - K_{mpc} x(k) \end{aligned} \tag{6.34}$$

where I_m and O_m are the identity and zero matrix with dimension $m \times m$ respectively.

6.8 MODEL PREDICTIVE CONTROL WITH CONSTRAINTS

The performance of a control system can significantly deteriorate when the control signals from the original control design meet the operating constraints [Wang, 2009d]. The performance degradation due to this problem can be reduced to a certain degree if the operational constraints can be introduced in the optimal control formulation.

The main idea in constrained control design is to modify the control variable ΔU to satisfy condition when the constraints become active. This could be done systematically in MPC control design through the optimisation process. Obviously, the unconstrained optimisation previously described in the previous section needs to be reformulated to incorporate constraints in the optimisation calculation.

There are three types of constraint variations typically used in control application: (1) constraints based on the incremental control variable, $\Delta u^{min} \leq \Delta u(k) \leq \Delta u^{max}$ (2) constraints based on the amplitude of control variable, $u^{min} \leq u(k) \leq u^{max}$, and (3) constraints based on output variable, $y^{min} \leq y(k) \leq y^{max}$. For a plant with multiple inputs and outputs, the constraints are specified for each input and output independently. For example:

$$\begin{aligned}
\Delta u_1^{min} &\leq \Delta u_1(k) \leq \Delta u_1^{max} \\
\Delta u_2^{min} &\leq \Delta u_2(k) \leq \Delta u_2^{max} \\
\Delta u_3^{min} &\leq \Delta u_3(k) \leq \Delta u_3^{max} \\
&\vdots \\
\Delta u_m^{min} &\leq \Delta u_m(k) \leq \Delta u_m^{max}
\end{aligned}$$

Since predictive control problem is formulated and solved in the framework of receding horizon, the constraints are taken into consideration for all future sampling instants. By taking example of constraints on incremental control variation, all constraints are expressed in the form of parameter vector ΔU as follows:

$$\Delta U = [\Delta u(k) \quad \Delta u(k+1) \quad \Delta u(k+2) \quad \cdots \quad \Delta u(k+N_c)]^T$$

The general statement of MPC optimisation problem with the effect of constraints involves finding the optimal control solution ΔU that minimises the cost function (6.31), which is reproduced here as follows:

$$J = (R_s - \Gamma x(k))^T (R_s - \Gamma x(k)) - 2\Delta U^T \Phi^T (R_s - \Gamma x(k)) + \Delta U^T (\Phi^T \Phi + \bar{R}) \Delta U$$

The minimisation of (6.31) is subject to inequality constraints:

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Delta U \leq \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \tag{6.35}$$

where the data matrices are:

$$M_1 = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix}; \quad N_1 = \begin{bmatrix} -U^{min} + C_1 u(k-1) \\ U^{max} - C_1 u(k-1) \end{bmatrix}$$

$$M_2 = \begin{bmatrix} -I \\ I \end{bmatrix}; \quad N_2 = \begin{bmatrix} -\Delta U^{min} \\ \Delta U^{max} \end{bmatrix}$$

$$M_3 = \begin{bmatrix} -\Phi \\ \Phi \end{bmatrix}; \quad N_3 = \begin{bmatrix} -Y^{min} + \Gamma x(k) \\ Y^{min} - \Gamma x(k) \end{bmatrix}$$

Considering a MIMO case, U^{min} and U^{max} are defined as column vectors with mN_c elements of u^{min} and u^{max} respectively. ΔU^{min} and ΔU^{max} are also defined as a column vectors with mN_c elements of Δu^{min} and Δu^{max} respectively. M_1 and N_1 are matrices associated with constraints on the amplitude of control variables, M_2 and N_2 are matrices associated with constraints on the increment of control variables, and the matrices M_3 and N_3 refer to constraints on output variable. In the case of manipulated

input variable constraints, matrices C_1 and C_2 are defined as follows:

$$C_1 = \begin{bmatrix} I \\ I \\ I \\ \vdots \\ I \end{bmatrix}; \qquad C_2 = \begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ I & I & 0 & \cdots & 0 \\ I & I & I & \cdots & 0 \\ \vdots & & & & \\ I & I & I & \cdots & 0 \end{bmatrix}$$

In the cost function (6.31), the matrix $\Phi^T\Phi + \bar{R}$ is also known as the Hessian matrix and is assumed to be positive definite. Since the cost function in Equation (6.31) is considered quadratic with linear inequality constraints, the problem of minimising the cost function becomes the equivalent problem for finding the optimal solution in Quadratic Programming (QP) study. The constraint equation in Equation (6.35) can be expressed in a compact form as:

$$M\Delta U \leq \gamma \tag{6.36}$$

where M is a matrix representing the constraints with its number of rows equal to the number of constraints and the number of column equal to the dimension of vector ΔU (if considering SISO case). When the system is fully imposed with all three types of constraints, the number of constraints are equal to $D = (4 \times m \times N_c) + (2 \times q \times N_p)$, where the constant m is the number of inputs and the constant q is the number of outputs in the system.

6.8.1 The Hildreth's Quadratic Programming Procedure

In the formulation of MPC problem, constraints imposed in the problem formulation represent the desired range of operation for the plant. The Active Set method, Primal Dual Inferior Point method, Hildreth's QP procedure and Shor's r-Algorithm are some examples of methods that handle optimisation solutions involving constraints [Wang, 2009d, Truong, 2007, Luenberger and Ye, 2008]. In the Active Set method, the active constraints (constraints that meet the condition $M\Delta U = \gamma$) need to be identified along

with optimal control decision variable ΔU. The Active Set method requires an iterative updating procedure to test the constraint conditions (active or inactive) before solving for optimal decision variable. The main drawback of the Active Set method is that it produces quite a high computation load if many constraints are imposed on the optimisation problem [Wang, 2009d, Truong, 2007].

The Primal-Dual method such as Hildreth's QP procedure can be used to overcome the computation burden of the Active Set method. Generally, the Primal-Dual method systematically identifies the inactive constraints and eliminated them directly in the solution [Wang, 2009d]. This would lead to a much simpler programming implementation for finding the optimal solution for the constrained control problem. To be consistent with QP literature notation, the cost function in Equation (6.31) and the associate constraints are reformulated as:

$$\begin{aligned} J &= \frac{1}{2}x^T Ex + x^T F \\ &\text{subject to} \quad Mx \leq \gamma \end{aligned} \tag{6.37}$$

where matrix E and F are compatible matrices and vectors in the quadratic programming problem in Equation (6.31). Variable x denotes the decision variable or the control decision variable ΔU. Matrix E is also known as the Hessian matrix with symmetric $mN_c \times mN_c$ dimension and F is a column vector with mN_c elements.

In order to minimise the objective function (6.31) subject to inequality constraints $Mx \leq \gamma$, the following Lagrange expression is considered:

$$J = \frac{1}{2}x^T Ex + x^T F + \lambda^T (Mx - \gamma) \tag{6.38}$$

where λ denotes the Lagrange Multiplier vector with a dimension equivalent to D number of constraints. The Equation (6.38) is also known as the primal problem in literature. The Lagrange Multiplier λ indicates whether a constraint is either active or inactive. By definition, the elements in Lagrange Multiplier vector are non-negative and if the element is $\lambda_i = 0$ then the i^{th} constraint is inactive, which indicates that a

solution has satisfied the constraint condition $Mx - \gamma < 0$. In contrast, if the elements in the Lagrange Multiplier is $\lambda_i > 0$, the corresponding constraint is active.

The estimation of the constraints can be obtained using a dual method as suggested in Wang and Young [2006]. Assuming there is a solution of ΔU that satisfy $Mx - \gamma < 0$, the dual problem to the original QP problem can be stated as follows:

$$\underset{\lambda \geq 0}{max}\ \underset{x}{min} \left[\frac{1}{2} x^T E x + x^T F + \lambda^T (Mx - \gamma) \right] \tag{6.39}$$

The minimisation over decision variable x is assumed unconstrained and the optimal solution is given as:

$$x = -E^{-1}F - E^{-1}M^T\lambda \tag{6.40}$$

By inserting Equation (6.40) into Equation (6.39), the dual problem can be written as:

$$\underset{\lambda \geq 0}{max} \left[-\frac{1}{2}\lambda^T L \lambda - \lambda^T K - \frac{1}{2} F^T E^{-1} F \right] \tag{6.41}$$

where the matrices L and K are given by:

$$L = ME^{-1}M^T \tag{6.42}$$

$$K = ME^{-1}F + \gamma \tag{6.43}$$

The Equation (6.41) is also a QP problem and is equivalent to:

$$\underset{\lambda \geq 0}{min} \left[\lambda^T L \lambda + \lambda^T K + \frac{1}{2} \gamma^T E^{-1} \gamma \right] \tag{6.44}$$

The set of optimal Lagrange multipliers that minimise the dual function in Equation (6.44) are denoted as λ^*. Using the value of λ^*, the decision variable x is obtained for the MPC control using the following formulation:

$$x = -E^{-1}F - E^{-1}M^T\lambda^* \tag{6.45}$$

In order to obtain λ^* that approximates Equation (6.39), the Hildreth's quadratic

programming procedure can be used to solve the dual problem. The vector λ^* is obtained by separately adjusting the component of λ_i to minimise the cost function. If we consider one complete cycle through the components to be one iteration k, the method can be expressed explicitly as:

$$\lambda_i^{k+1} = max(0, w_i^{k+1}) \tag{6.46}$$

where,

$$w_i^{k+1} = -\frac{1}{l_{ii}} \left[k_i + \sum_{j=1}^{i-1} l_{ij}\lambda_j^{k+1} + \sum_{j=i+1}^{n} l_{ij}\lambda_j^m \right] \tag{6.47}$$

where the scalar l_{ij} and l_{ii} are defined as the ij^{th} and the diagonal elements of matrix L respectively, while the scalar k_i is the i^{th} element in the vector K. The iteration of Hildreth's QP procedure converges monotonically to an optimal Lagrange multiplier λ^* over a finite number of iterations. The requirements for the dual variables λ^* to converge to a set of fixed values are based on the conditions that the active constraints are linearly independent and the number of the active constraints remain less or equal to the number of decision variables x [Wang, 2009c]. If these conditions are violated, the iteration will terminates when the loop reaches its predefined maximum value. By deleting the inactive constraints in the resulting λ^* vector obtained from the iteration, the optimal decision variables x can be calculated using Equation (6.45) together with the corresponding active constraint matrix M.

6.9 CONTROL ARCHITECTURES AND IMPLEMENTATION

The complete helicopter UAS control architecture can be partitioned into smaller control subsystems using the cascaded control approach [Valavanis, 2007, Sanders et al., 1998]. In this approach, the complete helicopter control problem is decomposed into two cascaded loops as shown in Figure 6.3. The main function of the inner loop control is to stabilise the helicopter attitude by controlling the helicopter actuator. The inner loop control receives reference signals from the outer control loop which operates at a slower rate compared with the inner loop's sampling rate. The outer loop control is

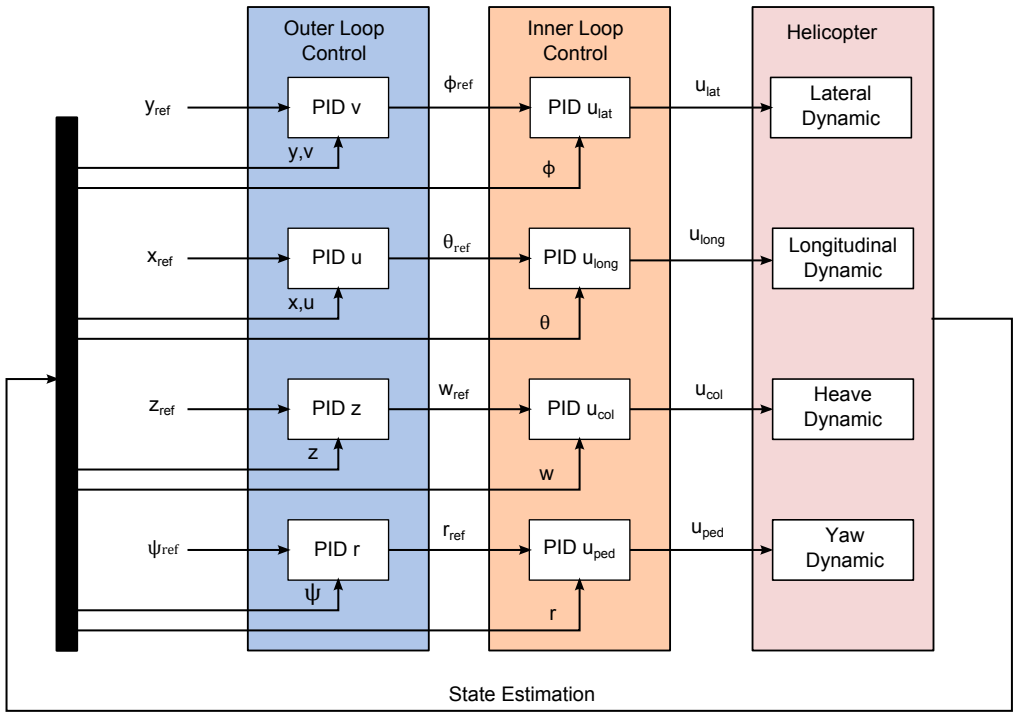


Figure 6.3 The helicopter control with cascaded control approach. Multiple SISO based PID controllers is used in the inner and outer loop.

responsible for guidance and generation of attitude or velocity commands for the inner loop control to achieve the desired position targets. In this study, the input/output pairing for the inner loop control is selected based on suggestions in Valavanis [2007] and Mettler [2003]. The suggested input/output pair is also expressed in Figure 6.3.

The overall control system architecture designed for the helicopter UAS consists of the control subsystems: a MIMO MPC with roll and pitch dynamics; a SISO MPC with yaw rate dynamics and a SISO MPC with altitude rate dynamics. The control system architecture is depicted in Figure 6.4. The control system architecture is selected as a TITO system considering coupling between lateral and longitudinal channels while yaw rate and altitude rate (heave) dynamics are treated separately [Mettler, 2003, Valavanis, 2007, Samal, 2009]. The arrangement of this controller architecture assumes that the tail rotor cyclic and the collective pitch do not influence the roll and pitch channel. In each of the control subsystems, a HMLP network is utilised to represent the dynamic model of the MPC. The training of these HMLP networks can be done using the off-line LM or recursive based GN training. In recursive based training, the initial weights of

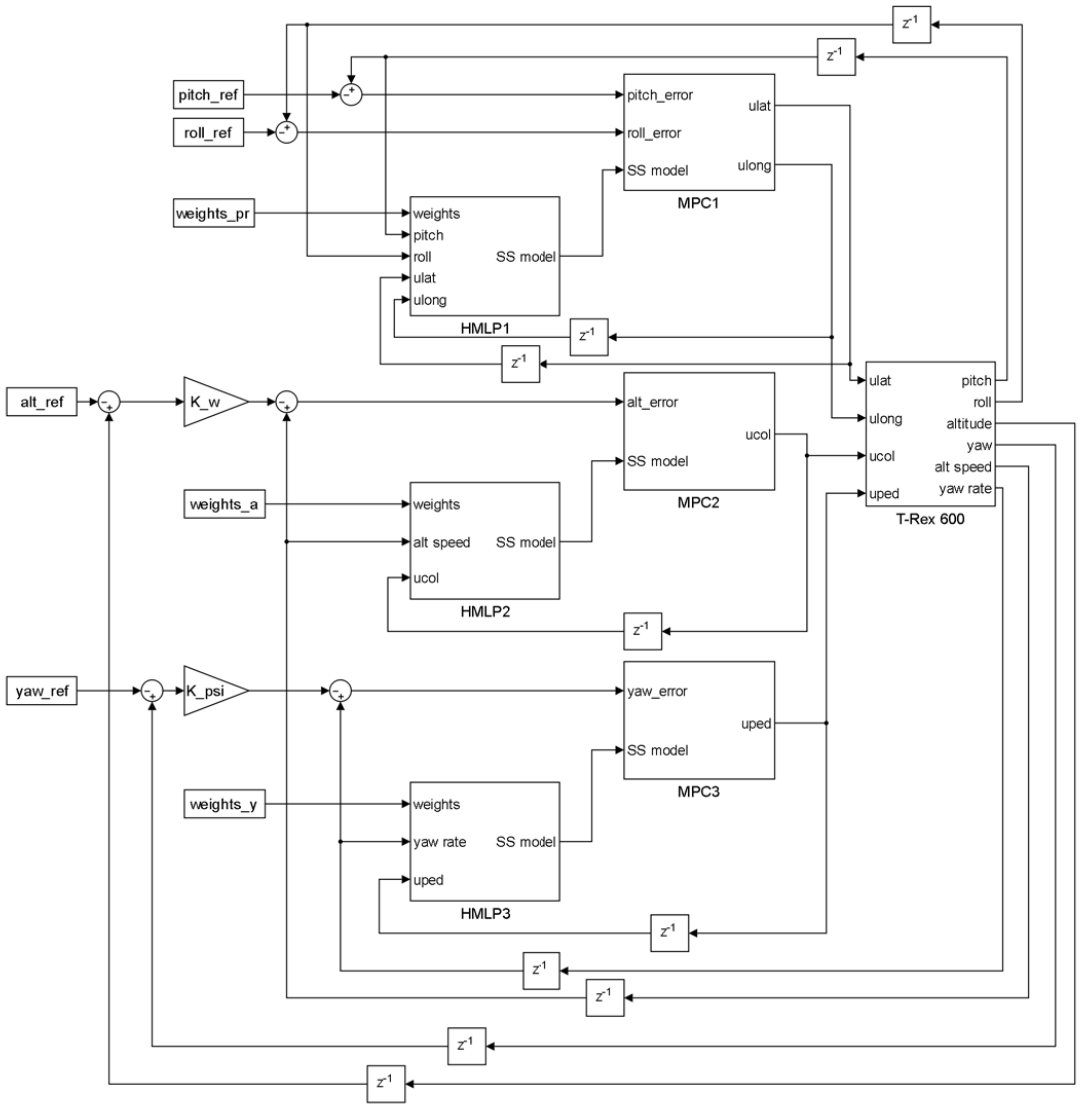


Figure 6.4 The unmanned helicopter control system architecture with NNAPC controller.

the HMLP network can be initialised using the weights trained with the off-line training. The time regressor vectors for each of these HMLP networks are constructed after obtaining pitch, roll and yaw angle, yaw rate and altitude rate measurement from the on-board sensors. Then, the prediction outputs from each HMLP network are fed to the instantaneous linearisation block to obtain the non-minimal state space models. Finally, all of the estimated linear models are transmitted to MPC loops for the optimisation iteration process before the new input signals are constructed and sent to the actuators.

The full implementation of the NNAPC control algorithm with recursive HMLP model is shown in the Figure 6.5. The control algorithm is implemented based on

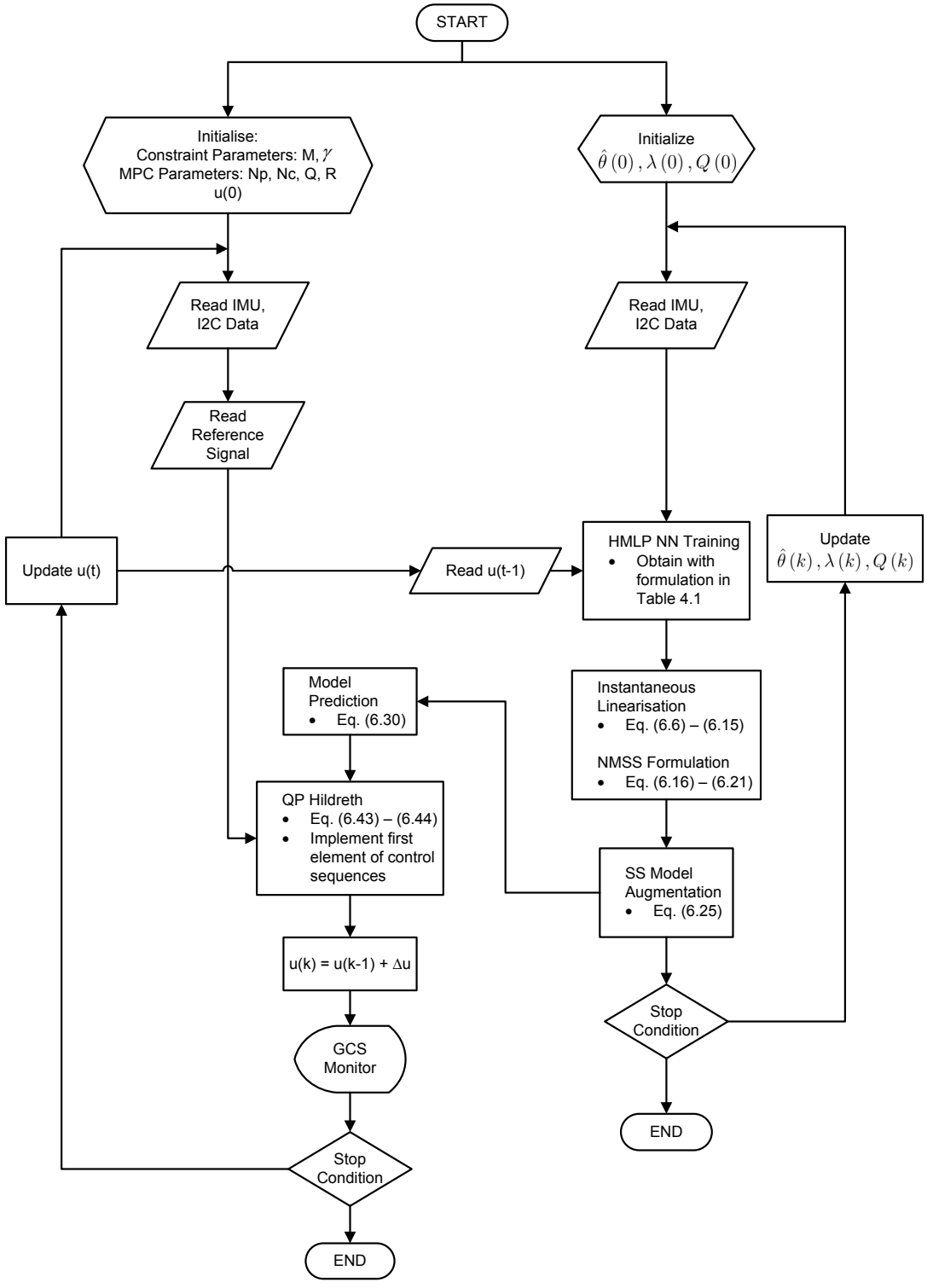


Figure 6.5 The NNAPC algorithm flowchart with recursive HMLP model. The training method shown in this figure is based on the recursive Gauss-Newton (rGN) method.

Labview® software from National Instrument. It starts with the initialisation of the NN and controller parameters. At the start of the iteration, the IMU (rotations) and I2C (altitude) modules collect data and produce the output measurement vector, $y(k)$. The HMLP module receives the previous calculated control input $u(k)$ and current output measurement $y(k)$ and constructs a linearised NMSS model. The NN model implemented in this algorithm can be obtained either from off-line training or recursive training. The model prediction module receives this state space model and predicts the future behaviour of the helicopter dynamics. This prediction is later on used in the Hildereth's quadratic programming to optimise the control problem objective function subject to specified constraints. Finally, the calculated control input is sent to the FPGA module to be executed. The calculated control input and the current output measurements are also sent to the ground control station (GCS) for monitoring purposes.

6.10 SUMMARY

In this chapter, the theoretical foundation of the NNAPC algorithm was presented. This algorithm was proposed in order to overcome the computational limitation of the non-linear NN based MPC. Different components of the NNAPC algorithm such as: the principle of instantaneous linearisation of the NN model, formulation of the NMSS and the augmented state space model, the prediction operation, optimisation and constraints handling of the MPC algorithm were discussed. In the next chapter, the implementation results of the proposed system identification method and control algorithm are presented to validate the feasibility of the methods for autonomous hovering flight.

Chapter 7

FLIGHT CONTROL SYSTEM DESIGN: RESULTS AND DISCUSSION

7.1 INTRODUCTION

This chapter presents the validation results of the proposed NN based predictive controller discussed in Chapter 6. The flight tests were carried out in the experimental test rig mentioned in Section 3.3, in order to validate the performance of the proposed system identification and control methods in autonomous hovering flight. This chapter describes the implementation of the experiments in Section 7.2, including the alteration of the software program to reduce computation time in Subsection 7.2.1. Section 7.3 presents the experimental results of the proposed flight controller. In Subsection 7.3.1, the tuning procedures are discussed and results on the effect of the tuning parameters are given for the NNAPC with the recursive NN training. After the best tuning parameters for each controller have been determined, the tracking performance for each controller is compared under parameter variations and discussed in Subsection 7.3.3. The robustness of controllers is also tested under input disturbance in this subsection. Finally, the chapter summary is given in Section 7.4.

7.2 FLIGHT TESTS IMPLEMENTATION

The automatic flight controller can be designed incrementally where the experiments are split in several development stages. The best tuning parameters obtained in these development stages are then used as the basic tuning parameters in the design of the hovering (4 DOF) flight controller. The pitch and roll dynamics can be tested together or

separately since both dynamic motions are quite slow compared with the yaw dynamic. This is due to the effect of stabiliser bar that introduces an increased damping effect to the rotor [Mettler, 2003]. The yaw dynamics is tested in a separate test because of the fast dynamic response while the altitude control is tested in the final part of the tuning refinement stages. Similar to the yaw control test, the altitude control is also done in a separate test since the involvement of chaotic movement induced by the sudden throttle change can severely damage the helicopter if not properly executed. This leads to the following experimental options:

1. Test the pitch rotation under automatic control while locking roll and yaw rotations and keeping zero altitude.

2. Test the pitch and roll rotation under automatic control while locking yaw rotation and keeping zero altitude.

3. Test the pitch and roll controller while manually controlling yaw rotation and varying altitude levels.

4. Test the pitch, roll and yaw controller while keeping zero altitude.

5. Test the altitude controller while locking pitch, roll and yaw rotations.

6. Complete 4 DOF controller test.

7.2.1 Computation Time Improvement

The core of the UAS control hardware is the NI sbRIO-9605 flight computer which provides a 400 MHz real-time processor with reconfigurable FPGA. The real-time sbRIO-9605 flight controller handles the sampling of the IMU measurement, NN training (or prediction), data logging and the MPC controller calculation. The data logging module is set at the lowest priority and sampling rate, while the NN and NNAPC module are run at a higher rate. Experience shows that setting the sampling rate of the NN training and NNAPC module at 30 Hz are sufficient for hover (4 DOF) control. However, in the early development phase of the flight controllers, the 4 DOF NNAPC controllers with

recursive NN training were found to be too computationally intensive to perform on the real-time processor.

Several improvements are made in the software program to reduce the total computation time. The main improvements are listed in Table 7.1. The first improvement is in the implementation of Xsens 'plug and play' IMU driver. The original IMU driver from Labview® software spent considerable computation time on determining the data format, number of bytes and redundant error handling. Since the data format is known, these functions can be removed. The same principle applies to the NMSS model conversion and the Hildreth's QP algorithm, where redundant functions are removed. Finally, a Look-Up-Table (LUT) design is implemented for calculation of Γ and Φ, to reduce the amount of calculation involved in the nested 'For' loops.

Table 7.1 The list of improvements made in the control software program. Note that the timing statistics are obtained using profiling tools in Labview®.

Type of improments	Sub-VI	ms/run (old)	ms/run (new)	% gain
Removed redundant functions from the IMU driver	Xsens Mti IMU Driver	114.15	4.29	96.2
Removed redundant calculation in the NMSS model	Model Conversion	1.73	0.35	79.8
Removed redundant calculation: constraint violation check	Hildreth's QP	1.33	1.07	19.6
LUT design for the model prediction gains	MPC Gains	16.02	4.08	74.5

Besides these improvements, several design tips were followed to keep the computation time as low as possible. These design tips are:

- The number of flight data samples transmitted to the ground control station is limited to 10 samples at a time to reduce network overhead.

- Graphical visualisation of the output variables is avoided in the real-time processor in order to remove unnecessary computational overhead. The visualisation of the sensor measurement and control input monitoring is displayed on the ground control station via the network shared variables protocol.

- The mathematical calculations that can be done without iterations or involve a predetermined matrix size are placed outside processing loops as much as possible to reduce the computation overhead.

- The debugging option is disabled in the program to reduce overhead.

- Input constraints are only imposed on the first sample of the control horizon to reduce computational efforts. There is no significant performance improvement obtained from imposing constraints on all control sequence as confirmed in Wang [2009a].

7.3 EXPERIMENTAL RESULTS

This section presents the control experimental results for the unmanned helicopter system. The first part of the experiments is conducted to identify and demonstrate the effect of tuning parameters on the NNAPC flight controller performance. The tuning parameters are identified only in the pitch, roll and yaw channels. The altitude controller uses the same tuning parameters obtained from the pitch, roll and yaw controller experiment. The tuning parameters comparison is done for control penalty factor r_w and prediction horizon length N_p. The control penalty factor r_w is determined first in the experiment and the best r_w value is then used for the N_p effect experiment. Table 7.2 lists the specification of the HMLP networks used in the implementation of the NNAPC controllers. The HMLP network is selected to model the helicopter's dynamics due to the network capability to model the dynamics with fewer weight connections compared with the standard MLP network.

Table 7.2 The HMLP network parameters used in the NNAPC controller implementation.

Network Parameters	Dynamic Channels		
	Coupled Roll-Pitch	Yaw	Altitude
Output(s)	ϕ, θ (rad)	r (rad)	w (m/s)
Regression Vector, φ	$\phi, \theta, \delta_{lat}, \delta_{lon}$	r, δ_{ped}	w, δ_{col}
Number of Past Outputs	2	2	2
Number of Past Inputs	1	1	1
Number of Hidden Neurons	2	2	2

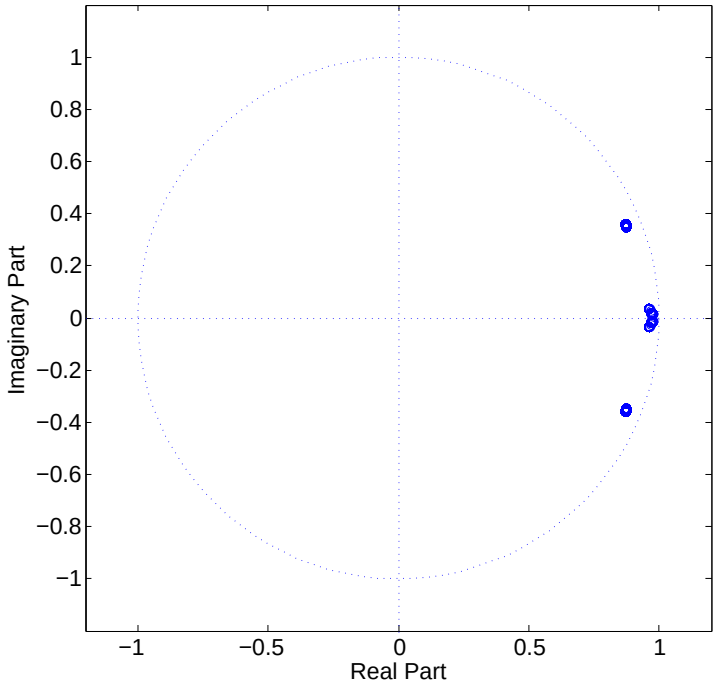


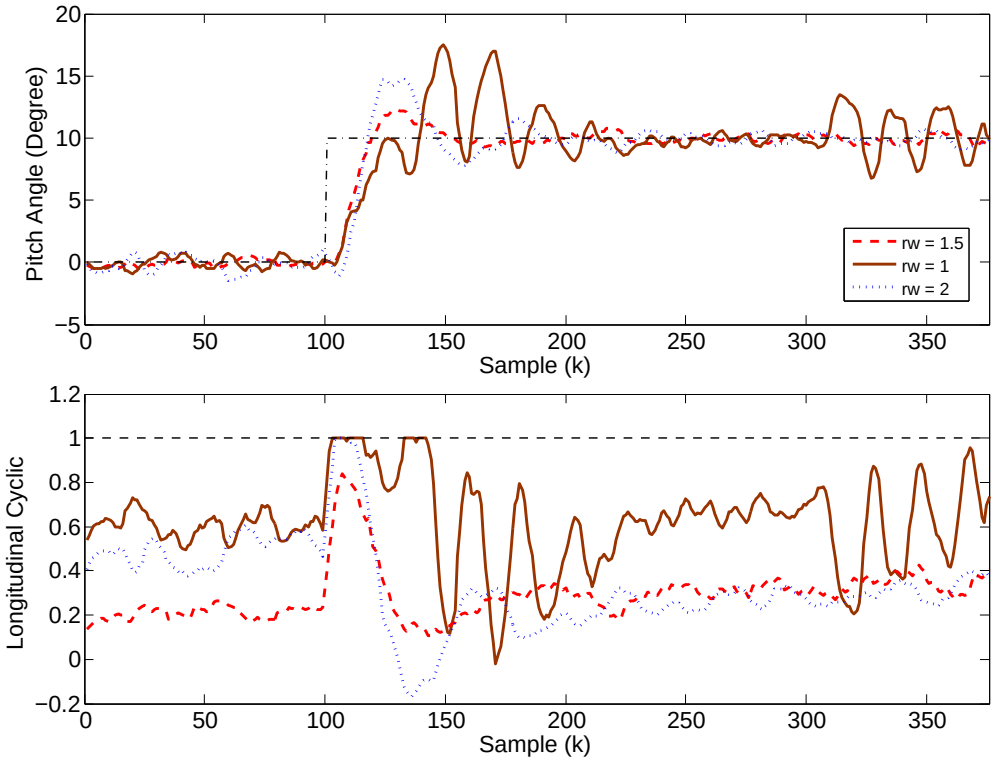
Figure 7.1 The location of poles of the extracted linear models from the instantaneous linearisation process. The poles were obtained at every simulation interval and are super-positions in the z-domain plot.

7.3.1 Flight Controller Tuning

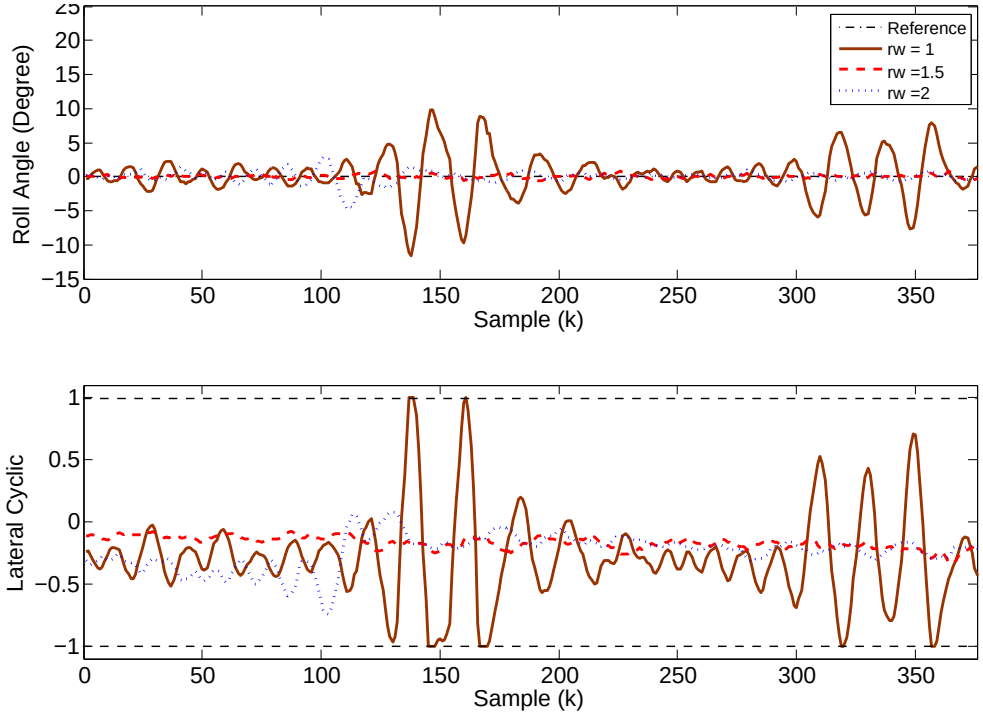
The proposed NNAPC design consists of three tuning parameters that need to be properly tuned to achieve good controller performance. The tuning parameters that influence the closed loop response of the system comprise the control horizon length N_c, the control penalty factor r_w and the prediction horizon length N_p. The control horizon value N_c can be selected to be equal to or exceed the number of output lag (number of past outputs) terms [Soeterboek, 1992]. Another way to determine the number of control horizon N_c steps is based on the number of unstable or poorly damped poles of the system [Norgaard, 2000, Soeterboek, 1992]. Figure 7.1 shows the pole-zero map of the coupled roll and pitch dynamic system of the unmanned helicopter system. The result in this figure is obtained using the linear models extracted from the instantaneous linearisation process at each sampling instant. In this study, the control horizon N_c is selected at 3 sample steps to exceed the number of poorly damped poles of the system. Nevertheless, a higher number of control horizon N_c can be selected for better controller performance but at the expense of increased computation time.

The response of the NNAPC controller with the recursive NN training is studied under the variation of r_w values. The NNAPC controller validation test is made under the coupled pitch-roll controller action. First, the helicopter is operated manually with the yaw axis servomechanism locked. The throttle is then increased gradually to the value of 0.25 as the initial rotor speed (Full transmitter's stick deflection is 1). Next, the controller action in pitch and roll is activated after the main rotor gains sufficient rotation speed. During this stage, the roll-pitch controller drives the helicopter to level orientation in the roll and pitch channels respectively. All other control channels such as the yaw and altitude channels are under manual control. The yaw servomechanism is then released after the straight and level condition is achieved. The throttle is then increased to 0.60 causing a small increase in the altitude channel. Next, a step response in pitch channel is activated, stepping up from 0° to 10° (positive pitch indicates nose down) while maintaining level orientation in the roll channel. The constraints on the amplitude of lateral and longitudinal cyclic inputs are set between $-1 \leq u(k) \leq 1$, while the prediction horizon length N_p is set to a sufficiently long value at 10. The step response is maintained for 275 samples, which equals approximately 9 s at a sample period time of 33 ms.

Figure 7.2 shows the result for comparing different values of r_w with the summary of controller performance given in Table 7.3. The MSE value is calculated in the period after the steady state value has been obtained. The settling time is calculated when the response settles within the 15 % settling time threshold, starting after initiation of the step response. The overshoot value is given as the percentage of the highest peak compared with the settling point, and the rise time is determined as the time taken by the response signal to move between 10 % and 90 % of the step value. The result shows that $r_w = 1.5$ produces the best response compared with other r_w values. When r_w is assigned with a low value, $r_w = 1$, the rate of change of the control signal u is not highly penalised in the cost function which results in an active controller response as can be seen in the results. The MSE for $r_w = 1$ is high compared with other values of r_w in both dynamic channels signalling that the controller produces an unstable response with poor compensation performance. Contrary to this, setting $r_w = 2$ means that controller



(a) Pitch angle θ



(b) Roll Angle ϕ

Figure 7.2 The control response comparison with $N_p = 10$ and different r_w values for the roll and pitch channels. The dash-dotted lines indicate the constraint limits imposed on the control input calculation.

Table 7.3 The control performance comparison with various r_w values.

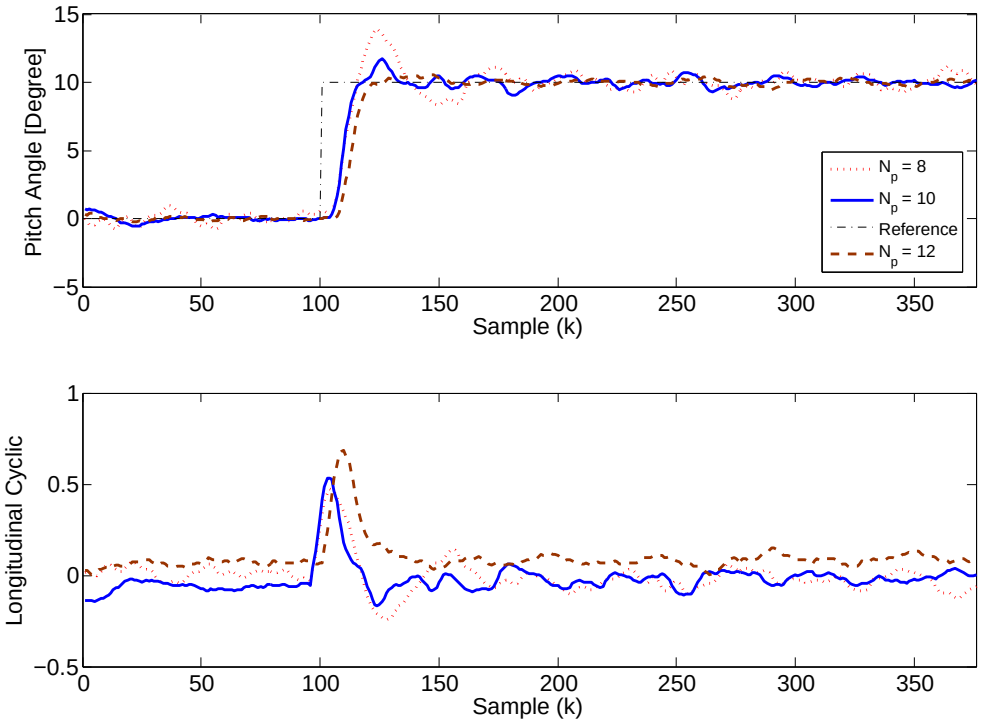
Tuning parameter		MSE	Settling Time (s)	Overshoot (%)	Rise Time (s)
θ	$r_w = 1.0$	1.9219	unstable	unstable	unstable
	$r_w = 1.5$	0.1934	1.32	23.02	0.36
	$r_w = 2.0$	0.1975	3.80	47.45	0.59
ϕ	$r_w = 1.0$	10.26	n/a	n/a	n/a
	$r_w = 1.5$	0.07	n/a	n/a	n/a
	$r_w = 2.0$	0.75	n/a	n/a	n/a

action is penalised too much in the cost function resulting in higher overshoot value.

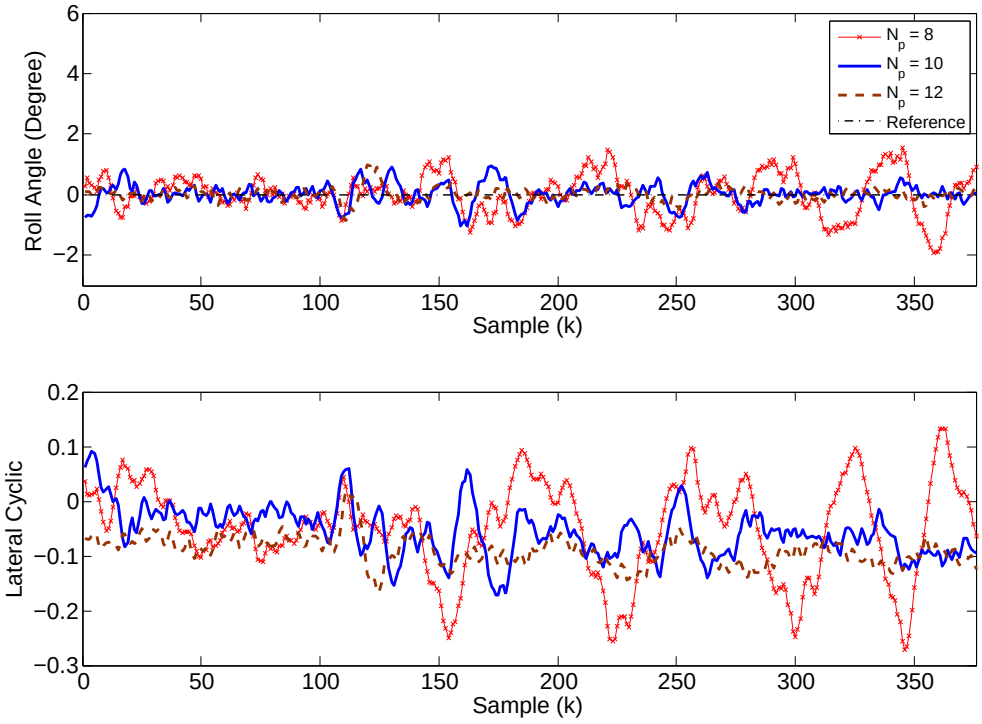
Next, the controllers' performance is tested under the effect of prediction horizon N_p variation. The experiment procedures for this test are similar to the previous experiment procedures. The only difference is that the test is performed under active roll, pitch and yaw controllers while the altitude channel is under manual control. Figure 7.3 and 7.4 show the results for comparing the controller performances under different values of N_p. Both of the figures show the controller performance results for roll, pitch and yaw channels. The control penalty factor of $r_w = 1.5$ is used in each of MPC controller setup. The same input constraints are used for roll and pitch controller as in the previous test, whereas, the constraint on the yaw rate controller is set up based on the rate of the control input, $-0.015 \leq \Delta u(k) \leq 0.015$.

Since the RC helicopter dynamics is unstable, several stability augmentation devices such as the mechanical stabiliser bar and active yaw rate feedback controller have been introduced to the RC helicopter system to enable human pilots to control the RC helicopter with ease. The stabiliser bar acts as a rate lagged feedback mechanism that dampens the roll and pitch control sensitivity due to the lateral and longitudinal cyclic inputs [Mettler, 2003]. The feedback mechanism would ease the manual control of the helicopter and at the same time reduce the destabilising effect on the helicopter due to wind gust or turbulence.

In addition, the yaw rate feedback controller and a gyro sensor are also included in all RC helicopter models to improve the stabilisation of the helicopter in the yaw axis. Figure 7.5 shows the illustration of the link between the yaw dynamics of the helicopter and the component of the stability augmentation systems. Ideally, the yaw



(a) Pitch angle θ



(b) Roll Angle ϕ

Figure 7.3 The NNAPC response comparison with $r_w = 1.5$ and various N_p values for roll and pitch channels.

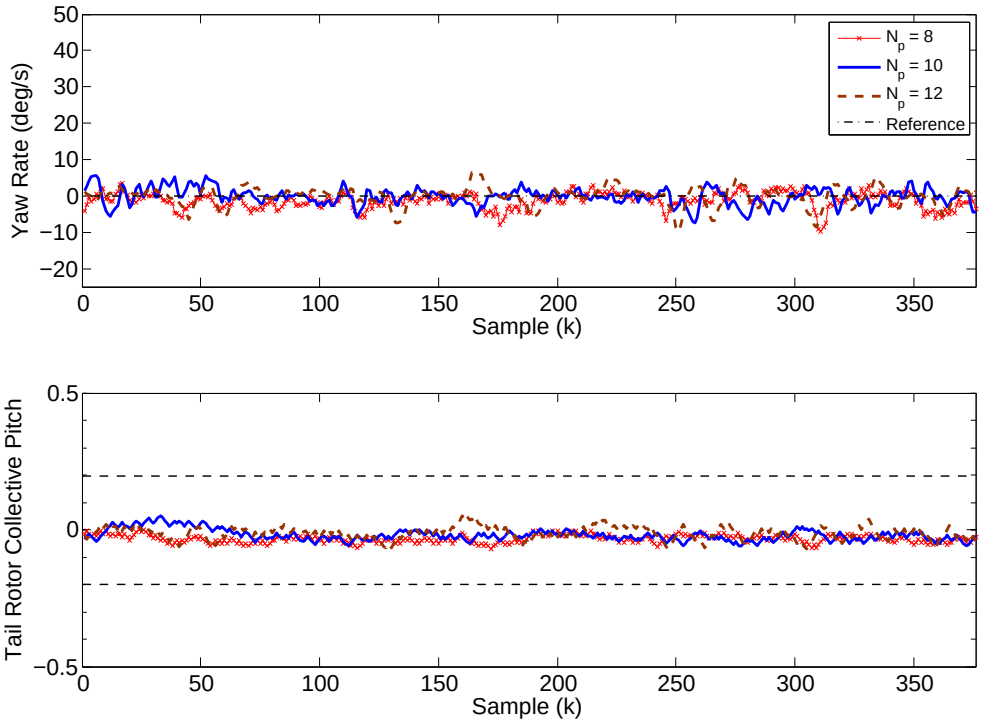


Figure 7.4 The NNAPC response comparison with $r_w = 1.5$ and various N_p values for the yaw channel.

rate feedback controller and the gyro sensor unit can be removed from the unmanned helicopter system since the autopilot system has utilised an IMU unit as the main attitude measurement and the NNAPC algorithm for the automatic yaw rate control. However, the gyro sensor and the commercial yaw rate feedback controller are kept in the unmanned helicopter system design to facilitate the manual control of the helicopter.

Incorporating the commercial yaw rate feedback controller into the AFCS introduces a new control signal that increases the existing control input calculation from the NNAPC controller. Experience shows that this kind of controller arrangement makes the unmanned helicopter turn aggressively to the right when the yaw servomechanism is released in the beginning of the experiment. The sudden movement happens due to the added control signal value from the conventional yaw rate feedback controller that makes the actual tail rotor input movement $\hat{\delta}_{ped}$ surge to the minimum deflection value (-1) in short period of time. In several runs, the unmanned helicopter could not even maintain the orientation in the yaw channel due to the tendency of the actual control input to get stuck at the minimum deflection value (-1). To prevent the helicopter

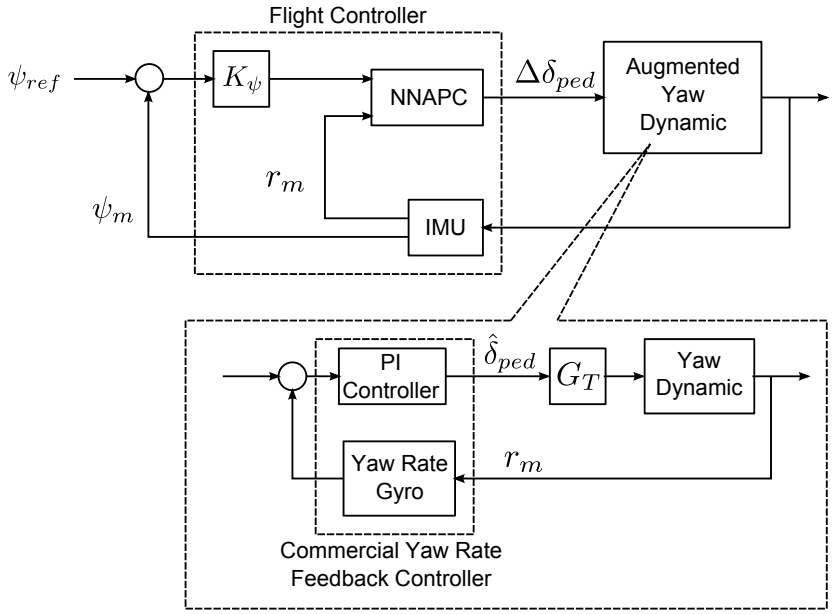


Figure 7.5 The block diagram showing the links between the on-board flight controller, the bare yaw dynamic and the components of augmented system in the yaw channel: the yaw rate gyro and the PI controller. G_T indicates the actuator gain for the actuator that controls the tail rotor pitch while K_ψ denotes the controller gain for yaw angle compensation.

from spinning out of control in the yaw axis, a saturation limit is introduced in the NNAPC controller calculation that bounds the calculated control action value between $-0.2 \rightarrow 0.2$.

Table 7.4 summaries the controller performance for this comparison using the MSE, settling time, overshoot and rise time performance indicators. The result shows that the controller with $N_p = 10$ and $N_p = 12$ produces better response compared with $N_p = 8$. The selection of prediction horizon length that is lower than $N_p = 8$ would result in an unstable closed loop response. The prediction horizon N_p parameter needs to be selected long enough to ensure that the closed loop system is stable and satisfies the control performance criteria [Maciejowski, 2002]. However, it is often impossible to choose a prediction horizon that predicts too long into the future as the solution can increase the computation time of the NNAPC controller. Moreover, the NNAPC optimisation is based on the prediction from the extracted linear models which are only valid in the linearisation window. Thus, the prediction from these linear models is not sufficiently accurate for prediction in the far future. Several alternative methods that can be used to approximate the N_p value are based on the rise time or dead time information of the

dynamic system [Shridhar and Cooper, 1997, Clarke et al., 1987, Garriga and Soroush, 2010]. However, the empirical tuning approach is the most preferred method since the tuning is based on the actual control performance which gives a better assignment of N_p value compared with the approximation method.

Table 7.4 The control performance comparison with various N_p values.

Tuning parameter		MSE	Settling Time (s)	Overshoot (%)	Rise Time (s)
θ	$N_p = 8$	0.40	1.78	39.20	0.26
	$N_p = 10$	0.16	0.84	17.05	0.25
	$N_p = 12$	0.06	0.54	4.25	0.34
ϕ	$N_p = 8$	0.46	n/a	n/a	n/a
	$N_p = 10$	0.12	n/a	n/a	n/a
	$N_p = 12$	0.05	n/a	n/a	n/a
r	$N_p = 8$	6.51	n/a	n/a	n/a
	$N_p = 10$	5.50	n/a	n/a	n/a
	$N_p = 12$	7.12	n/a	n/a	n/a

The tuning results for the yaw angle control are given in Figure 7.6 and Table 7.5. The yaw angle control is achieved by giving $r_{ref} = K_\psi (\psi_{ref} - \psi_m)$ as a reference trajectory to the NNAPC yaw rate controller. The variables ψ_{ref} and ψ_m denote the reference and the measured yaw angle respectively. In this test, the automatic controller actions are deactivated for the roll and pitch channels while allowing only the automatic yaw rate controller to compensate for the trajectory error in the yaw channel. The servomechanisms are also locked for the pitch and roll channels. The yaw controller performance is tuned with $r_w = 1.5$ and $N_p = 10$ during this test. The result obtained suggests that the gain value of $K_\psi = 0.5$ gives the lowest rise time with high MSE value, which indicates a high steady state error in the system response. The yaw controller gives better compensation performance and faster response as the gain value is increased to $K_\psi = 1$.

Table 7.5 The control performance comparison with various K_ψ values.

Tuning parameter		MSE	Settling Time (s)	Overshoot (%)	Rise Time (s)
ψ	$K_\psi = 0.5$	11.20	3.73	1.46	5.02
	$K_\psi = 0.75$	8.92	2.48	1.61	2.16
	$K_\psi = 1.0$	1.86	1.83	1.31	1.48

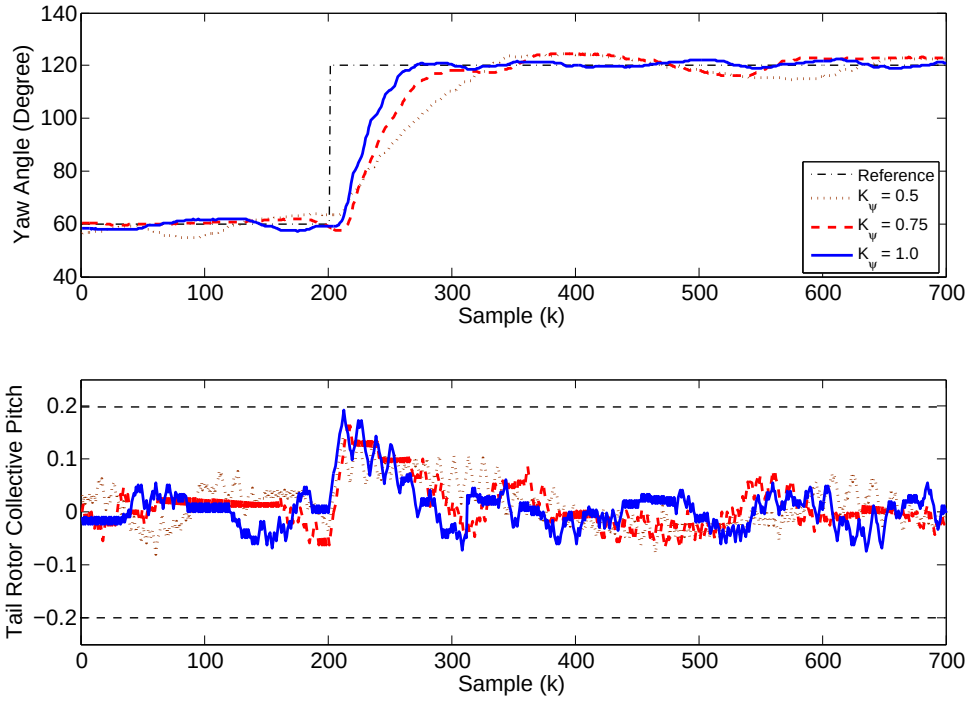


Figure 7.6 The NNAPC response comparison with various K_ψ values for the yaw angle compensation. The prediction horizon $N_p = 10$ and control penalty factor $r_w = 1.5$ are selected for this controller with constraints on control input rate $-0.015 \leq \Delta u(k) \leq 0.015$.

7.3.2 4-DOF Controller

The optimal controller parameters determined from the previous flight tests are used for controlling four of the helicopter control channels simultaneously under hovering flight. The final tuning parameters for the NNAPC controller with the recursive NN training are given in Table 7.6. The experiment is performed in the following steps: Initially, the helicopter is placed in the start-up position with yaw rotation axis locked using the servomechanism. Then, the throttle command of the helicopter is increased gradually from 0 to 0.25, in order to gain enough main rotor speed. The controller actions for the pitch, roll and yaw channels are then activated to drive the helicopter around level orientations in pitch and roll channels, while the yaw controller tracks a constant zero angular speed in yaw motion. After the level flight condition is obtained, the yaw rotation locking mechanism is then released to allow the helicopter to move freely in the yaw axis. The throttle is then increased to create enough lift, which is determined around 0.65. Next, the altitude controller is activated to drive the helicopter to track a 10 cm altitude reference. Figure 7.7 shows the hovering controller developed

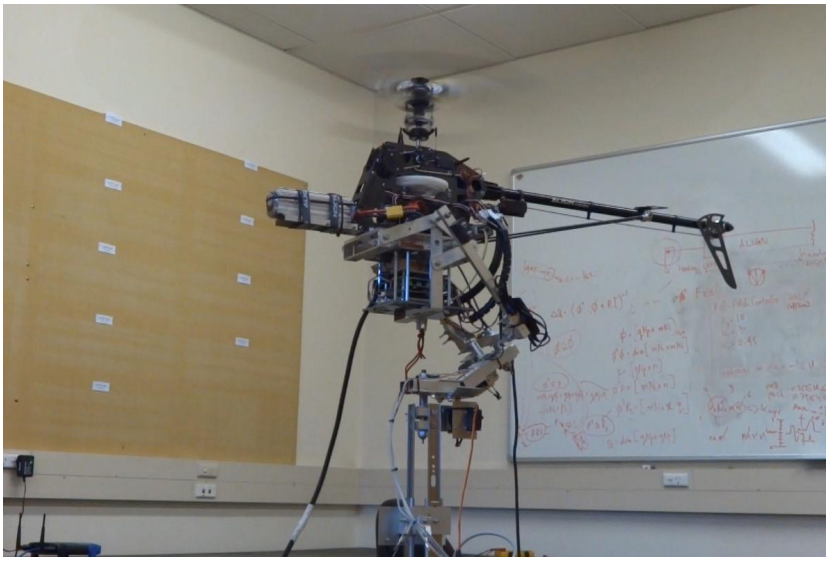


Figure 7.7 The 4 DOF NNAPC controller with recursive NN training in action.

Table 7.6 The final controller settings for the hovering flight test.

Tuning Parameters	Symbol	Value
State Cost	Q	C^TC
Control Cost	r_w	1.5
Control Horizon	N_c	3
Prediction Horizon	N_p	10
Roll-Pitch Amplitude Constraints	$u^{max,min}$	$\pm$ 1
Yaw/Altitude Rate Constraints	$\Delta u^{max,min}$	$\pm$ 0.01
Yaw Rate Gain	K_Ψ	1.00
Altitude Speed Gain	K_w	0.75

in action. The step response is maintained for ∼600 samples, which equals 19.8 s at a sample period time of 33 ms.

Figure 7.8 and 7.9 show the output responses of the 4 DOF controllers in term of the pitch angle, roll angle, yaw rate and altitude speed, together with the corresponding control signals. Table 7.7 summarises the performance indicators of the controller responses. The MSE value is calculated in the period after the steady state condition is obtained. The settling time is calculated after initiation of the step response with settling threshold of 15 %. Again, the overshoot is determined as the percentage of the highest peak compared with the settling point, and the rise time is calculated as the time taken by the response signal to rise between 10 % and 90 % of the step input value. It can be seen from visual inspection of the controller responses and the performance

indicators that a full 4 DOF controller is realised using the proposed controller approach with good compensation performance in all four motion axes.

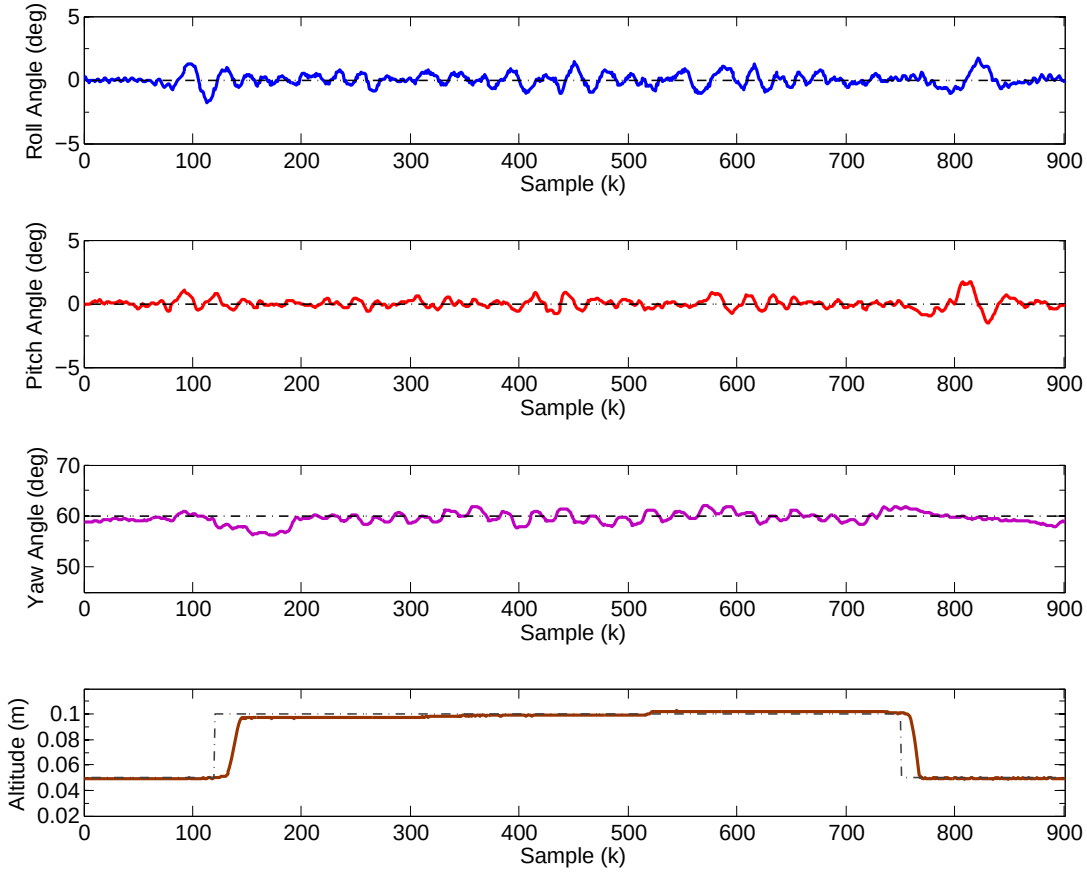


Figure 7.8 The control responses of the 4 DOF NNAPC controller with recursive NN training during hovering flight test.

Table 7.7 The 4 DOF NNAPC controller response under hovering flight.

Outputs	MSE	Settling Time (s)	Overshoot (%)	Rise Time (s)
Pitch Angle, θ	0.17	n/a	n/a	n/a
Roll angle, ϕ	0.25	n/a	n/a	n/a
Yaw angle, ψ	1.54	n/a	n/a	n/a
Altitude, z	4.10×10^{-6}	0.76	1.69	0.38

7.3.3 Flight Controller Performance

In this subsection, the NNAPC controllers developed in this work are further tested under the effect of parameter variations and input disturbances. There are several parameters that can vary in the helicopter dynamics such as the changes in the payload

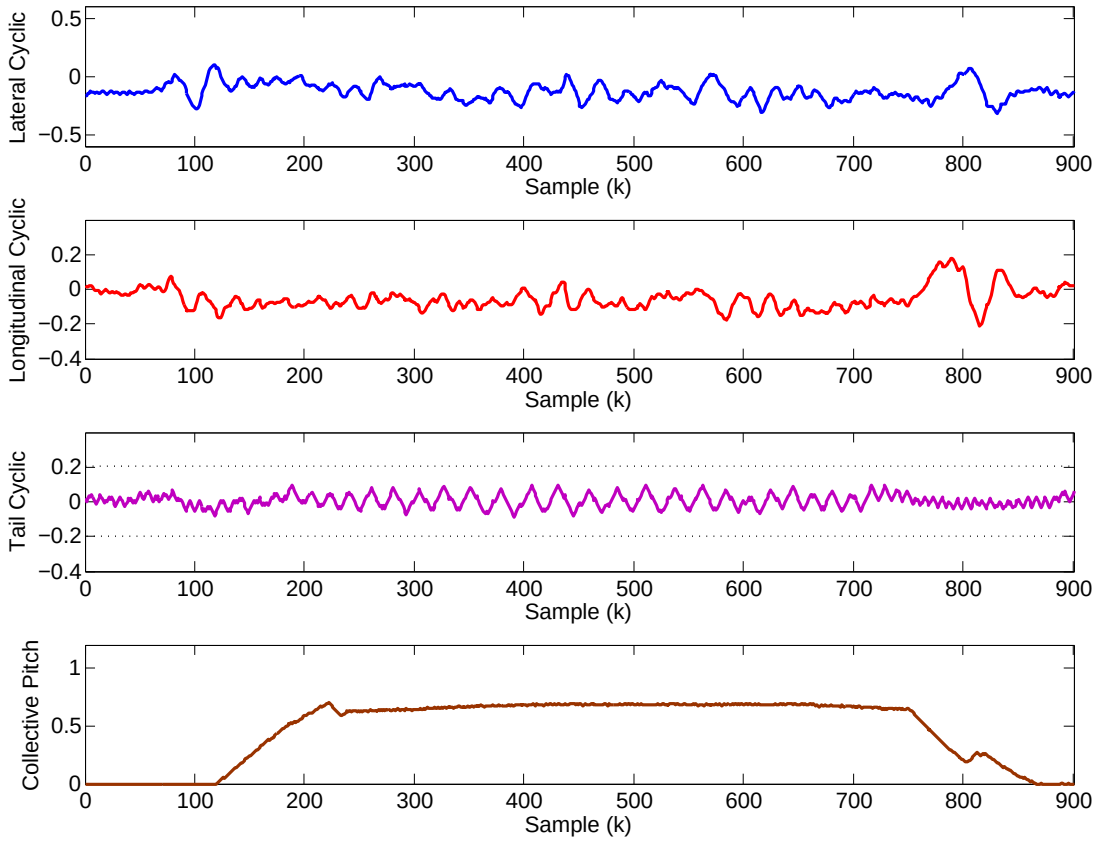


Figure 7.9 The corresponding control signals from the 4 DOF NNAPC controller with recursive NN training during hovering flight test.

weight, pitch and throttle curve setting or changes in the lift and drag forces acting on the helicopter during different flight conditions [Samal, 2009]. It is important for the flight controller design to perform adequately under these parameter variations. The parameter variation test carried out by implementing the NNAPC controllers in the roll, pitch and yaw channels. Two types of NNAPC controllers are used for comparison in the test, i.e. the NNAPC controller with the off-line trained NN model (the NN model obtained from off-line LM training or repeated rGN training) and the NNAPC controller with the recursive NN training (rGN). For easy reference to the controllers, the general term of 'NNAPC-Offline' and 'NNAPC-Online' will be used to represent the NNAPC controller types respectively. The compensation performance of the controllers is tested under different pitch and throttle curve setting from the one used in the NN training. The value changes made to the original pitch and throttle curves are shown in Figure 7.10. The collective pitch setting is created with lower throttle values in low and

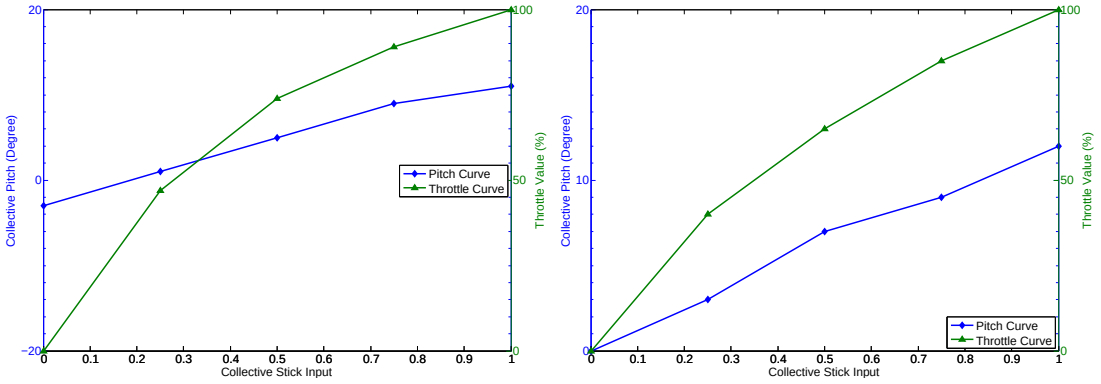
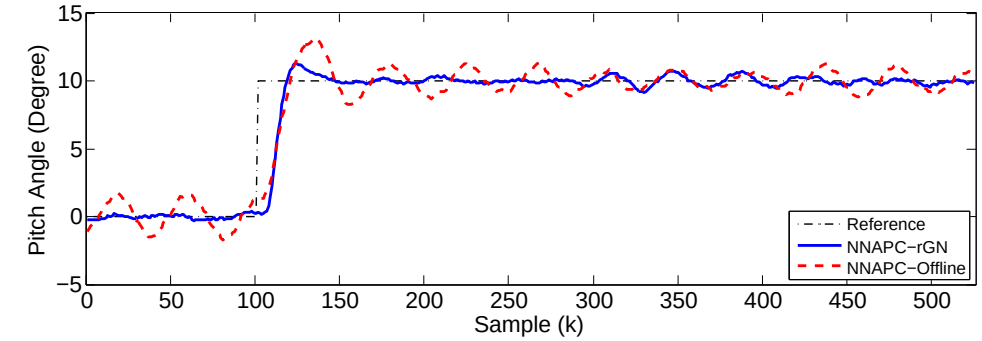


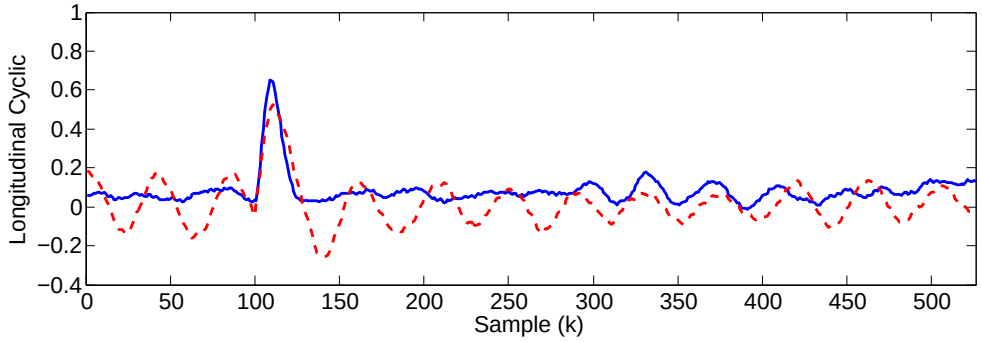
Figure 7.10 The parameter variation in collective pitch setting. (a) The original collective pitch and throttle setting from the RC transmitter which was used in the off-line NN training phase. (b) The new collective and throttle setting for controller comparison test under parameter variation.

mid range of the collective stick movement, while the new pitch curve is made slightly higher than the original value in mid range.

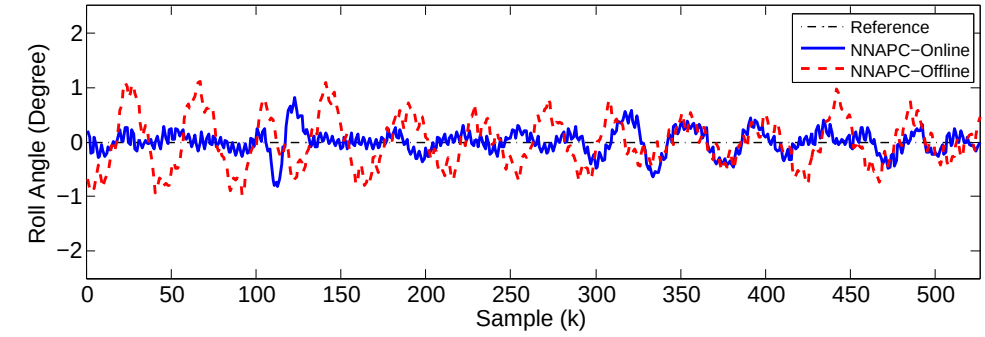
The NNAPC controller comparison results under the collective pitch and throttle setting changes are given in Figure 7.11 and Figure 7.12 with the overall performance indicators summarised in Table 7.8. As shown in the performance indicators, the NNAPC-Online controller outperforms the NNAPC-Offline controller in terms of the compensation error and the step response performance. In contrast to the NNAPC-Online controller, the NNAPC-Offline controller exhibits induced oscillation while tracking the given reference either in the roll or pitch axis. The obtained empirical finding validates that the NNAPC-Online controller performs well under the new parameter settings compared with the NNAPC-Offline controller. This is due to the fact that the NNAPC controller suffers performance degradation if an off-line NN model is used in the MPC prediction process. Since the NN model is not trained with these changes in the collective curve and throttle setting, a significant error due to model mismatch may occur and this modelling error impacts the accuracy of the MPC controller prediction, thus contributing to the decreased performance as indicated in Figure 7.11 and 7.12.

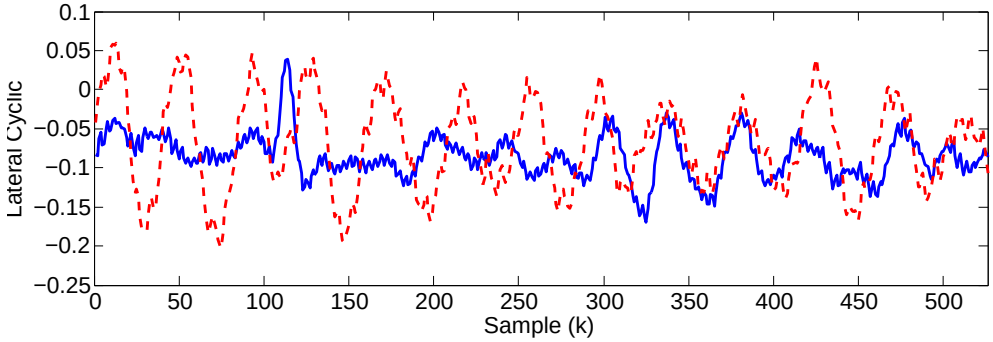
The same performance degradation also happens if the same flight controller tests are conducted under the changes of helicopter payload. In this test, a total of 1 kg of counterbalance weights in the test rig are reduced to increase the amount of payload that





(a) Pitch angle θ





(b) Roll Angle ϕ

Figure 7.11 The coupled roll-pitch controllers' response comparison under changes in the collective pitch and throttle curve settings.

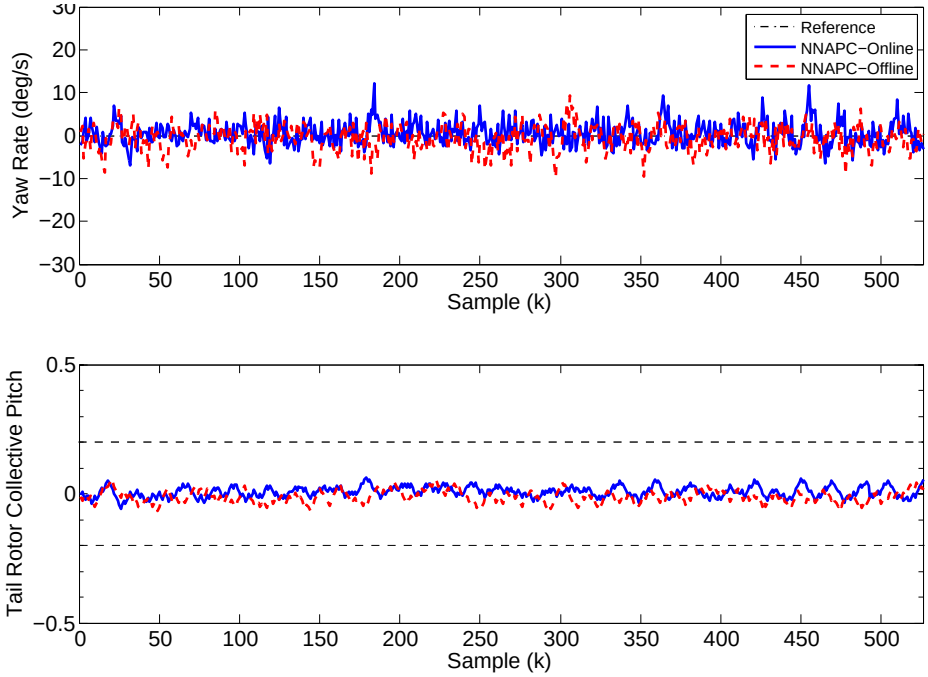


Figure 7.12 The yaw controllers response comparison under changes in the collective pitch and throttle curve settings.

Table 7.8 The controllers' performance comparison under changes in the collective pitch and throttle curve settings.

Tuning parameter		MSE	Settling Time (s)	Overshoot (%)	Rise Time (s)
θ	NNAPC-Online	0.08	1.00	13.52	0.32
	NNAPC-Offline	0.29	14.16	26.57	3.91
ϕ	NNAPC-Online	0.05	n/a	n/a	n/a
	NNAPC-Offline	0.21	n/a	n/a	n/a
r	NNAPC-Online	8.19	n/a	n/a	n/a
	NNAPC-Offline	9.34	n/a	n/a	n/a

the helicopter carries. The compensation response comparison between the controllers under consideration is given in Figure 7.13 with the performance analysis for the test given in Table 7.9. The NN weights used in the NNAPC-Offline controller is trained based on the data measured with the original counterbalance weight (5 kg). The same NN weights obtained in the off-line training are also used in the NNAPC-Online controller as the network's initial weights. The NNAPC-Online controller is found to be superior in terms of compensation performance compared with the NNAPC-Offline controller. An overshoot about 16.41% occurs in the NNAPC-Online controller response due to the inertia effect of the counterbalance weight. Even with the changes in counterbalance

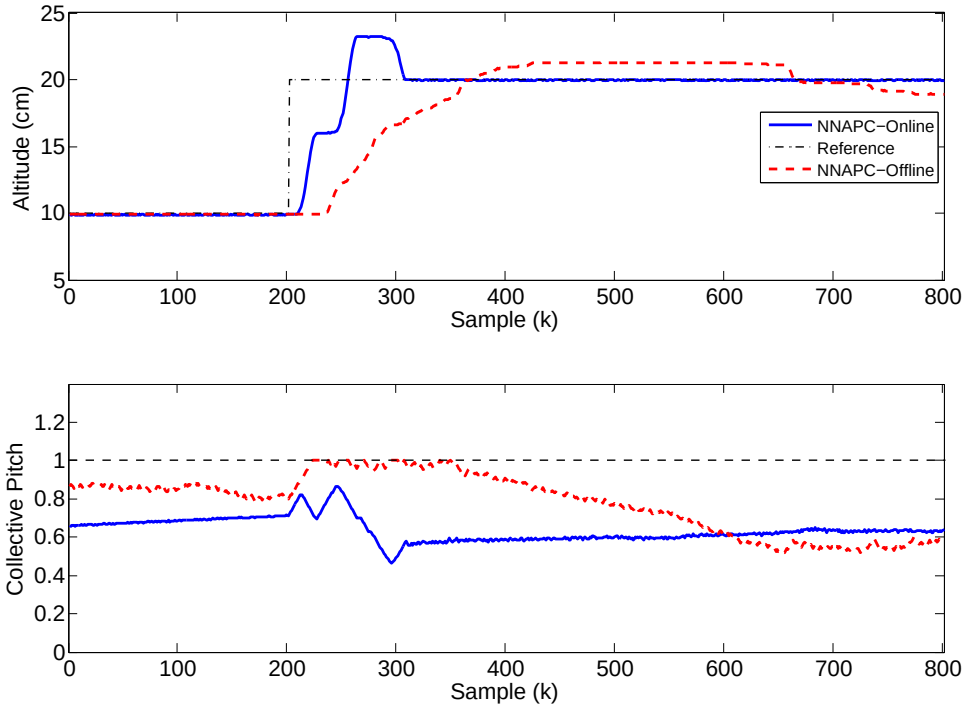


Figure 7.13 The altitude controllers' response comparison under the changes in test rig's counterbalance weights. The dash-dotted line indicates the constraint limits imposed on the control input calculation.

Table 7.9 The altitude controllers' performance comparison under changes in the counterbalance weight.

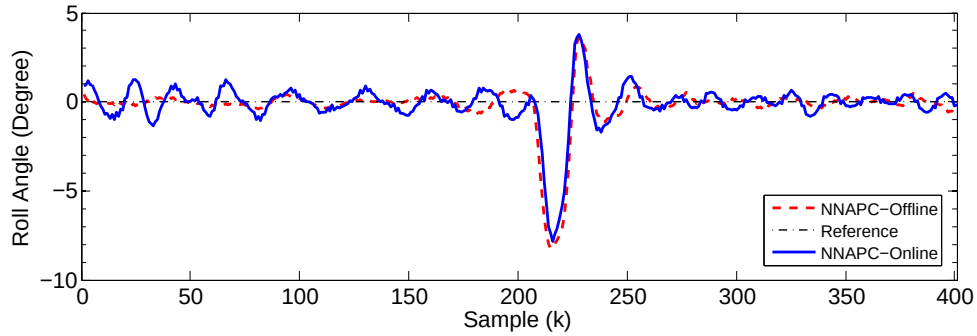
Tuning parameter		MSE	Settling Time (s)	Overshoot (%)	Rise Time (s)
z	NNAPC-Online	0.01	3.644	16.41	1.385
	NNAPC-Offline	0.53	16.72	12.88	3.25

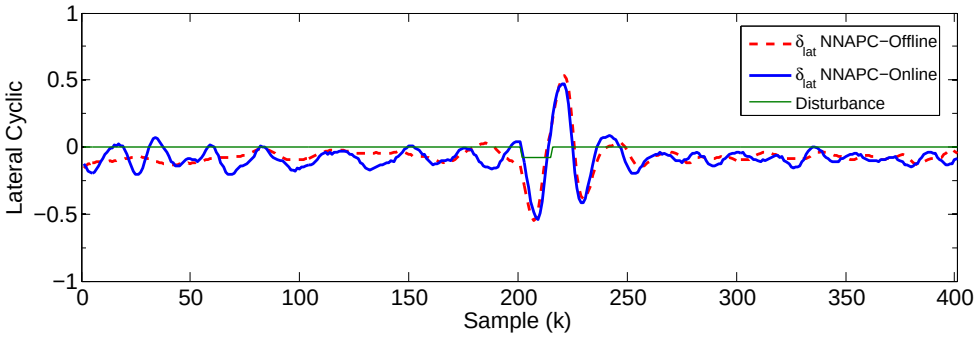
weight, the controller is able to reformulate the appropriate control action to bring back the system to the reference point. On the other hand, the NNAPC-Offline controller produced higher rise time and did not settle well in respect to the reference trajectory.

Next, the proposed NNAPC controllers are tested under the effect of input disturbance in each of the control channels. Introducing the disturbance effects to the calculated control inputs can be considered akin to the effect of wind gusts on the aerodynamic characteristics of the main rotor [Mettler, 2003]. It is important to test the controller compensation performance under input disturbances, in order to verify whether the proposed controller approach is able to operate in windy or disturbed conditions such as when the unmanned helicopter is flying in close proximity to structures. In this test, the helicopter is commanded to hover at 10 cm altitude reference, while

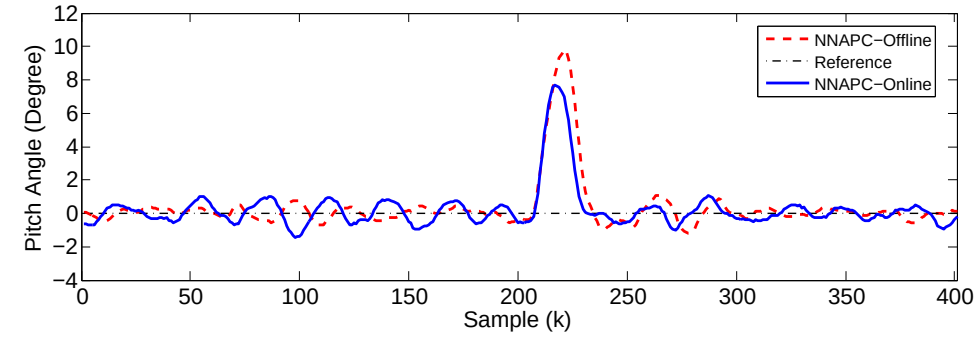
maintaining the level orientation (0°) in roll and pitch axes. The yaw controller is tasked to maintain a zero angular rate reference. To simulate the efficiency of the NNAPC controllers in rejecting the input disturbances, the controller input values are added with constant input disturbances at each control channel. These input disturbances are introduced one a time in every control channel after the steady state condition is achieved in each control channel.

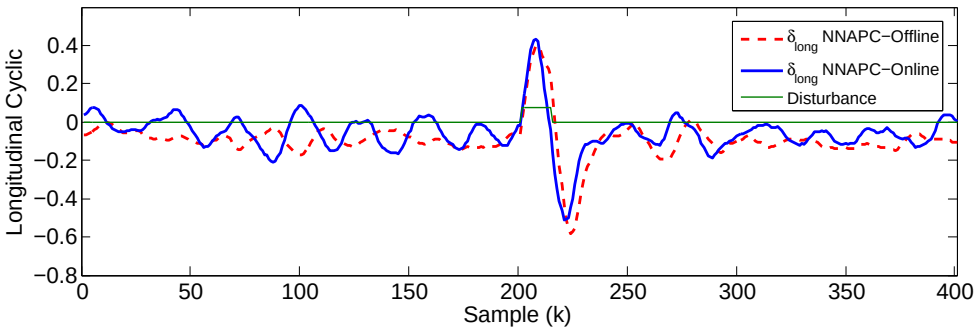
Figure 7.14 and 7.15 show the controller response profiles when the constant input disturbances of -0.075, 0.075, 0.02 and -0.01 are introduced to disturb the control moves $(\delta_{lat}, \delta_{long}, \delta_{ped}, \delta_{col})$ for a duration of 500 ms. It is shown that the controllers perform satisfactorily well under input disturbance effects with both controllers returned back to the reference conditions after the insertion of the disturbances within 2.8 s (86 samples) at the most. This demonstrates the robustness of the proposed NNAPC algorithms in rejecting sudden movement of the helicopter. Slight improvements can be seen for the control response of the NNAPC-Online controllers in the pitch and yaw axes where the control response returns to the reference value faster than the NNAPC-Offline controllers in 0.56 s (17 samples) and 0.26 s (8 samples) respectively.





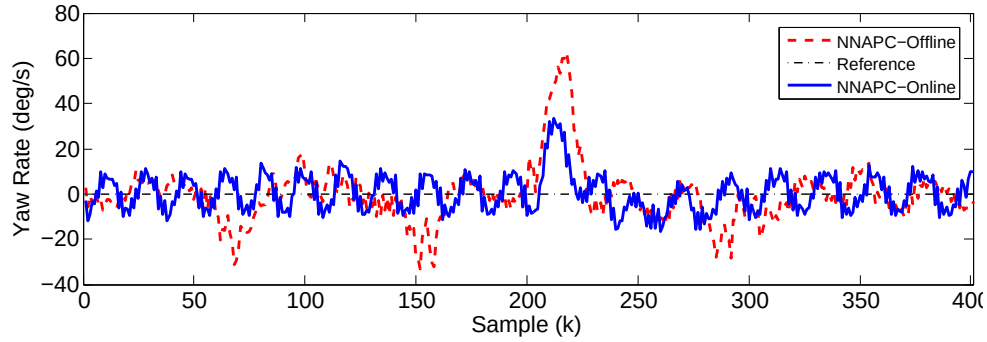
(a) Roll angle, ϕ

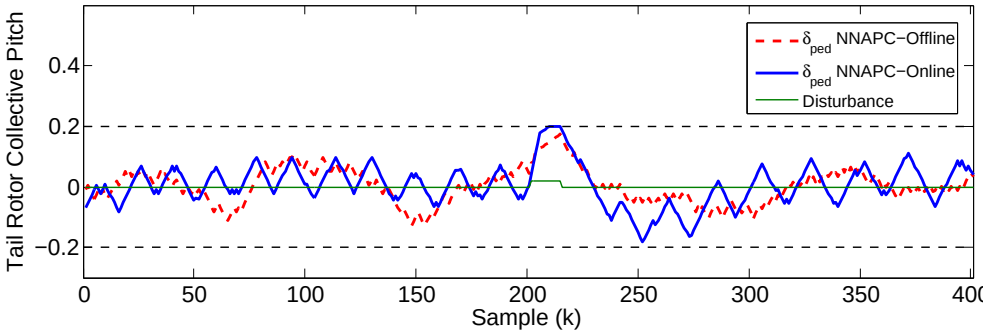




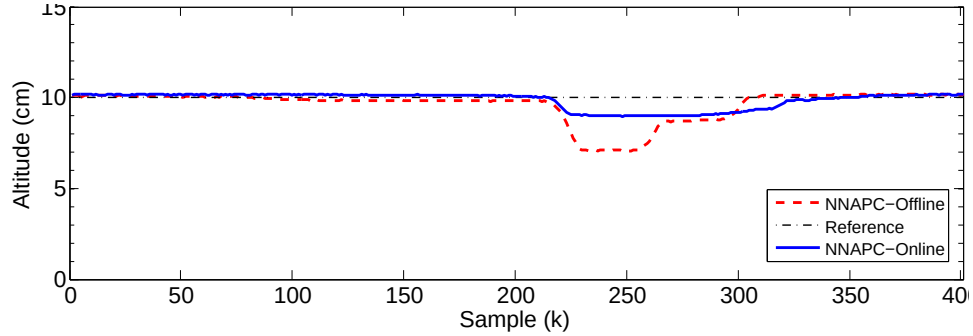
(b) Pitch angle, θ

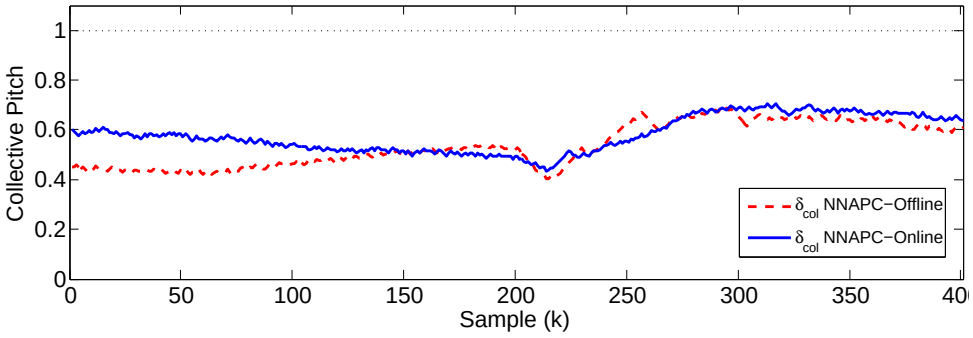
Figure 7.14 The coupled roll-pitch NNAPC controllers' response comparison under input disturbances. These disturbances are introduced one at a time with the control responses shown here being overlaps of two separate test runs.





(a) Yaw rate, r





(b) Altitude, z

Figure 7.15 The NNAPC controllers' response comparison under input disturbances for yaw and altitude channels. These disturbances are introduced one at a time with the control responses shown here being overlaps of two separate test runs.

7.4 SUMMARY

The flight test results for the unmanned helicopter system controlled by multiple NNAPC controllers are presented in this chapter. The automatic hovering flight controllers are designed according to the TITO architecture which result in three NNAPC controllers i.e. the coupled roll-pitch controller, the yaw rate controller and the altitude speed controller. Several flight tests have been conducted in the early parts of controller development to identified the control tuning parameters such as r_w, N_p, K_Ψ and K_w that would produced satisfactory control performance. The HMLP network presented in the Subsection 4.2.2 is used in the NNAPC controller prediction due the network capability to model the flight dynamics with fewer weight connections compared with the standard MLP network. The control tuning results show that tuning parameters such as $r_w = 1.5$, $N_p = 10$, $K_\Psi = 1.0$ and $K_w = 0.75$ produce the best flight control response compared with other tuning setting. Using the tuning parameters identified from the control tuning tests, a full 4 DOF NNAPC controller is successfully realised in the hovering flight condition with good compensation performance. In the hovering flight test, the unmanned helicopter is commanded to track a step response in altitude for a duration of 19.8 s while stabilising the helicopter in roll, pitch and yaw channels.

After the successful implementation of the 4 DOF flight controllers, the NNAPC controllers are further tested under the physical parameter variations that involve changes in the collective pitch curve setting, throttle curve setting and the test rig's counterbalance weight. The changes in the counterbalance weight can be equated to the changes in the unmanned helicopter payload during flight. Similar to the approach taken by Samal [2009], the changes in both collective and throttle curves of the helicopter blades are used as the test reference. In real flight condition, the total lift and drag forces acting on the helicopter vary during different flight and environment conditions. This can be treated similarly in the indoor test by setting changes in the collective pitch and throttle setting during flight. The performance of the NNAPC-Offline controller is then compared with the NNAPC-Online controller under parameter variation tests. The off-line NN model and the initial model for recursive HMLP are obtained using the

data measured from the previous parameter settings before changes is made. From the findings of the above studies, it can be concluded that the recursive NN model improved the NNAPC-Offline controller performance under the new parameter variations. This is due to the ability of the recursive NN model to track the time varying parameters of the helicopter dynamics. Since the off-line NN model is not trained with the new changes in physical parameter values, a significant model mismatch can occurs and this would affect the accuracy of the NNAPC controller's prediction, thus contributing to the deteriorating performance of the NNAPC-Offline controller.

Finally, the performance of the NNAPC controllers was further tested under input disturbances after tracking the given references. The proposed NNAPC approaches with either offline or online NN model were found to be efficient in compensating the effect of the input disturbances. Performance comparison between the NNAPC-Online and NNAPC-Offline controllers shows that the NNAPC-Online controllers offer improvement in regulating the input disturbance effects, particularly in the pitch and yaw axes. Despite the reported effectiveness of the NNAPC-Online controller in handling parameter variations and input disturbance effects, the good performance of the NNAPC-Online controller was obtained at the expense of increased computational load compared with the NNAPC-Offline scheme. However, this is justifiable for such a critical and challenging dynamic process. The computational load of the NNAPC-Online scheme can be implemented within given computational resources if appropriate selection of tuning options is made by users. Nevertheless, several suggestions are made in the next chapter on several options to improve the execution speed of the NNAPC-Online control scheme.

Chapter 8

CONCLUSIONS AND FUTURE WORKS

In this thesis, the system identification and control algorithms based on NN methodology were developed to control an unmanned helicopter system in hovering flight condition. The proven ability of the NN based system identification and controller design approach to adapt to the dynamic system changes allows for the development of a flexible and self-tunable flight controller that can be deployed into different UAS airframes in shorter development time. Furthermore, the fast and simpler development of the dynamic model using the NN approach offers significant benefit which reduces the cost associated with the development of large aerodynamic databases. The proposed system identification and NNAPC flight controllers for autonomous hovering were implemented and tested in simulation environment and indoor flight test facility.

A detailed overview of the unmanned helicopter platform as well as the avionics hardware used was provided. To prevent fatal crashes or damages to the helicopter system due to probable hardware failures or programming mistakes, a novel safety test rig was developed to restrict the movement of the helicopter during testing. The development of the safety test rig allows the helicopter system to move freely in 6 DOF flight. The test rig also was equipped with necessary instrumentation that provides the end users with position and attitude information of the helicopter.

Several NN based system identification algorithms were developed to model the unmanned helicopter dynamics from the flight test data. Three types of NN architectures were deployed to model the dynamics of the helicopter; namely the MLP, HMLP and modified Elman networks. The nearly optimal or optimal model structures for the proposed NN architectures can be found based on the proposed model validation tests.

The HMLP and modified Elman networks were found capable of providing prediction quality similar to the standard MLP network. The HMLP and modified Elman networks are recommended for modelling of the non-linear helicopter dynamics due to their smaller model structure which would subsequently reduce the computation load and training time of the NN model. Furthermore, the application of the modified Elman network in system identification procedures could simplify the overall NN system identification process, since the model structure of the Elman network does not need to be pre-determined. Two types of training algorithms were proposed in this study (off-line LM and recursive GN algorithms) to determine the weights of the NN models. The training algorithms based on the on-line training were used to improve the performance of the off-line training in terms of the generalisation and adaptability of the NN models under changing dynamic properties or uncertainties in NN model structure selection.

The theoretical foundation of the proposed NNAPC algorithm was presented and discussed in this work. The NNAPC flight controller was proposed in order to overcome the computational limitation of the non-linear NN based MPC. Different components of the NNAPC algorithm are highlighted in this work such as: the principle of instantaneous linearisation of the NN model, formulation of the NMSS and the augmented state space model, the prediction operation, optimisation and constraints handling of the MPC algorithm. The proposed NNAPC controller with NN model predictions are then validated in the developed test rig to achieved autonomous hovering of the unmanned helicopter system. Multiple NNAPC controllers were designed for each dynamic channel of the helicopter system and the results proved the robustness of the proposed NNAPC controller with recursive NN model training in handling variations in flight conditions and physical changes. The performance of the NNAPC controllers was also found to be satisfactory in the presence under input disturbances.

8.1 FUTURE WORKS

The current dynamics modelling simulation and flight experimental findings show promising results for designing an autonomous unmanned helicopter system. There is strong potential for this project to be extended to industrial applications as intended. These

recommendations are meant as future guidelines for those working on the continuation of the project. The recommendations are split into two parts, i.e. the improvement on test frame, and finally recommendation on the control system.

Several recommendations on the test rig design improvement are listed in particular order as follows:

1. The U-bent section on the upper part structure of the test frame needs to be replaced with a better mechanism design to allow easier roll movement of the helicopter without affecting the yaw rotation.

2. The caster wheels on the SCARA arms need to be replaced before implementing/testing the horizontal translational movement control. It is a bit difficult to rotate the caster wheels when the helicopter changes direction, thereby imposing unwanted dynamics on the helicopter flight. The caster wheels are recommended to be replaced with Omni wheels that can slide laterally with ease. Other alternatives such as fitting the SCARA arm with ball transfer units can also be considered to improve the translational movement of the helicopter.

3. To prevent cable twist issues, the current design needs to have additional mechanical constraints to limit yaw rotation.

Based on the flight validation of the NN based system identification and the approximate model predictive controller for autonomous hovering, the identification methods and controller scheme have potential to be used on various UAV platforms. However, the performance of the controller can be further improved. Some of the possible improvements are listed below:

1. The current system employs a single MIMO controller architecture for the pitch and roll channels with two separate SISO (Single-Input-Single-Output) controller architectures for the yaw and altitude channels. It is possible to improve the NNAPC controller performances using a single MIMO type controller as the main controller to handle the regularisation problem of the helicopter control. However, the lack of real-time processor computational resources prevented the

implementation of a full MIMO based controller. One of the primary causes of the computational restriction was due to the Labview® *plug and play* driver for MTi Xsens IMU which was executed together with other time critical processes such as the NN training and NNAPC controller processes. For example, if the user increases the number of prediction or control horizon steps too high, it would significantly impact the IMU measurement loop rate. More computational resources are used to maintain the loop timing of these time critical processes and this significantly reduces the IMU measurement loop rate, thus making the IMU produce chaotic results with measurement error. It is advisable to switch the data handling of the IMU measurement from the sbRIO's real-time processor to FPGA, in order to reduce the total computation burden of the sbRIO real-time processor.

2. Since the NN model is linearised continuously, the system dynamics can change rapidly at every time instance. For systems with non smooth non-linearity, there will be problems since the extracted linear models are generally valid near the operating point. To counter this problem and to gain a more stable system model, a low-pass filter can be implemented to the A-matrix in the State Space Model as recommended in Witt et al. [2007].

3. The translational controllers can be designed for the unmanned helicopter system, thus establishing a full six degrees-of-freedom controller. A recommendation for the design of translational control is to use several PID controllers, which will construct set-points for the inner loop roll-pitch MPC controller to drive the helicopter to the required translational positions.

4. To make the unmanned helicopter system ready for free flight, the altitude sensor needs to be replaced. It is recommended to combine two different positional measurement systems such as GPS for tracking translational motion or by using pressure sensor such as a barometer for accurate altitude measurement.

REFERENCES

A. Ab Wahab, R. Mamat, and S.S. Shamsudin. The effectiveness of pole placement method in control system design for an autonomous helicopter model in hovering flight. *International Journal of Integrated Engineering (Issue on Electrical and Electronic Engineering)*, 1(3):33–46, 2009.

N. Abas, A. Legowo, and R. Akmeliawati. Parameter identification of an autonomous quadrotor. In *Mechatronics (ICOM), 2011 4th International Conference On*, pages 1–8, May 2011. doi: 10.1109/ICOM.2011.5937198.

M. Agarwal. A systematic classification of neural-network-based control. *Control Systems, IEEE*, 17(2):75–93, 1997.

S M Ahmad, M H Shaheed, A J Chipperfield, and M O Tokhi. Non-linear modelling of a one-degree-of-freedom twin-rotor multi-input multi-output system using radial basis function networks. *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, 216(4):197–208, 2002. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=894926`.

Zheng A. Allgower, F. *Nonlinear model predictive control*, volume 26. Birkhauser, Boston, 2000.

O. Amidi. *An Autonomous Vision Guided Helicopter*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1996.

O. Amidi, T. Kanade, and K. Fujita. A visual odometer for autonomous helicopter flight. *Robotics and Autonomous Systems*, 28:185–193, 1999. doi: 10.1016/S0921-8890(99)00016-0.

Duarte Antunes, Carlos Silvestre, and Rita Cunha. On the design of multi-rate tracking controllers: Application to rotorcraft guidance and control. *International Journal of Robust and Nonlinear Control*, 20(16):1879–1902, 2010. ISSN 1099-1239. doi: 10.1002/rnc.1557. URL `http://dx.doi.org/10.1002/rnc.1557`.

V. S. Asirvadam. Adaptive regularizer for recursive neural network training algorithms. In *11th IEEE International Conferenc Computational Science and Engineering Workshops CSEWORKSHOPS '08*, pages 89–94, 2008.

S. N. Balakrishnan and R. D. Weil. Neurocontrol: A literature survey. *Mathematical and Computer Modelling*, 23(1-2):101–117, 1996. doi: 10.1016/0895-7177(95)00221-9.

P. Baldi. Computing with arrays of bell-shaped and sigmoid functions. In *Proceedings of the 1990 conference on Advances in neural information processing systems 3*, pages 735–742, Denver, CO, 1990. Morgan Kaufmann Publishers Inc.

B. W. Bequette. Non-linear model predictive control: A personal retrospective. *The Canadian Journal of Chemical Engineering*, 85(4):408–415, 2007.

M. Bergerman, O. Amidi, J.R. Miller, N. Vallidis, and T. Dudek. Cascaded position and heading control of a robotic helicopter. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 135 –140, 29 October - 2 November 2007 2007. doi: 10.1109/IROS.2007.4399450.

Gerald J. Bierman. *Factorization methods for discrete sequential estimation*, volume 128. Academic Press, New York, 1977.

S. A. Billings, H. B. Jamaluddin, and S. Chen. A comparison of the backpropagation and recursive prediction error algorithms for training neural networks. *Mechanical Systems and Signal Processing*, 5(3):233–255, 1991.

S. A. Billings, H. B. Jamaluddin, and S. Chen. Properties of neural networks with applications to modelling non-linear dynamical systems. *International Journal of Control*, 55(1):193 – 224, 1992.

Morten Bisgaard. *Modeling, Estimation, and Control of Helicopter Slung Load System*. PhD thesis, Department of Electronic Systems, Aalborg University, 2007.

A. R. S. Bramwell, George Taylor Sutton Done, and David Balmford. *Bramwell's helicopter dynamics*. American Institute of Aeronautics and Astronautics; Butterworth-Heinemann, Reston, VA, Jordan Hill, Oxford, UK, 2nd edition, 2001. 00049381 A.R.S. Bramwell, George Done, David Balmford. Helicopter dynamics ill. ; 25 cm. Rev. ed. of: Helicopter dynamics. c1976. Includes bibliographical references and index.

A. Budiyono, Yoon Kwang Joon, and F. D. Daniel. Integrated identification modeling of rotorcraft-based unmanned aerial vehicle. In *17th Mediterranean Conference on Control and Automation*, pages 898–903, 2009.

G. Buskey, G. Wyeth, and J. Roberts. Autonomous helicopter hover using an artificial neural network. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1635 – 1640, 2001. doi: 10.1109/ROBOT.2001.932845.

Guowei Cai, Ben Chen, and Tong Lee. An overview on development of miniature unmanned rotorcraft systems. *Frontiers of Electrical and Electronic Engineering in China*, 5:1–14, 2010. ISSN 1673-3460.

Guowei Cai, Biao Wang, Ben M. Chen, and Tong H. Lee. Design and implementation of a flight control system for an unmanned rotorcraft using rpt control approach. *Asian Journal of Control*, 15(1):95–119, 2013. ISSN 1934-6093. doi: 10.1002/asjc.504. URL `http://dx.doi.org/10.1002/asjc.504`.

A.J. Calise and R.T. Rysdyk. Nonlinear adaptive flight control using neural networks. *Control Systems, IEEE*, 18(6):14 –25, dec 1998. ISSN 1066-033X. doi: 10.1109/37.736008.

E. F. Camacho and C. Bordons. *Model predictive control*. Advanced textbooks in control and signal processing. Springer, London ; Berlin, 2004.

Eduardo Camacho and Carlos Bordons. Nonlinear model predictive control: An introductory review. In Rolf Findeisen, Frank Allgwer, and Lorenz Biegler, editors, *Assessment and Future Directions of Nonlinear Model Predictive Control*, volume 358 of *Lecture Notes in Control and Information Sciences*, pages 1–16. Springer Berlin / Heidelberg, 2007. ISBN 978-3-540-72698-2. URL `http://dx.doi.org/10.1007/978-3-540-72699-9_1`.

C. L. Castillo, W. Moreno, and K. P. Valavanis. Unmanned helicopter waypoint trajectory tracking using model predictive control. In *Mediterranean Conference on Control and Automation (MED '07)*, pages 1–8, 2007.

M. Castillo-Effen, C. Castillo, W. Moreno, and K.P. Valavanis. Control fundamentals of small / miniature helicopters - a survey. In KimonP. Valavanis, editor, *Advances in Unmanned Aerial Vehicles*, volume 33 of *Intelligent Systems, Control and Automation: Science and Engineering*, pages 73–118. Springer Netherlands, 2007. ISBN 978-1-4020-6113-4. doi: 10.1007/978-1-4020-6114-1_4. URL `http://dx.doi.org/10.1007/978-1-4020-6114-1_4`.

Juan Andrade Cetto, Jean-Louis Ferrier, Joaquim Filipe, Nikos Vitzilaios, and Nikos Tsourveloudis. *Safe Test Flights for Small Rotorcrafts*, volume 37 of *Lecture Notes in Electrical Engineering*, pages 153–166. Springer Berlin Heidelberg, 2009.

S. Chen, C. F. N. Cowan, S. A. Billings, and P. M. Grant. Parallel recursive prediction error algorithm for training layered neural networks. *International Journal of Control*, 51(6):1215–1228, 1990.

S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *Neural Networks, IEEE Transactions on*, 2(2):302 –309, 1991. ISSN 1045-9227.

Yahya Chetouani. Use of a neural-network-based approach for a reliable modelling of a distillation column. *International Journal of Modelling, Identification and Control*, 11(1/2):71–78, 2010.

Girish Chowdhary and Ravindra Jategaonkar. Aerodynamic parameter estimation from flight data applying extended and unscented kalman filter. *Aerospace Science and Technology*, 14(2):106–117, 2010.

Marco La Civita. *Integrated modeling and robust control for full-envelope flight of robotic helicopters*. PhD thesis, Carnegie Mellon University, 2003. 936907 Adviser - William Messner Adviser - Takeo Kanade.

D. W. Clarke, C. Mohtadi, and P. S. Tuffs. Generalized predictive control–part i. the basic algorithm. *Automatica*, 23(2):137–148, 1987. doi: 10.1016/0005-1098(87)90087-2.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2:303–314, 1989.

Olurotimi A. Dahunsi and Jimoh O. Pedro. Neural network-based identification and approximate predictive control of a servo-hydraulic vehicle suspension system. *Engineering Letters*, 18(4):357–368, 2010.

Konstantinos Dalamagkidis, Kimon Valavanis, and Les Piegl. Autonomous autorotation of unmanned rotorcraft using nonlinear model predictive control. *Journal of Intelligent and Robotic Systems*, 57(1):351–369, 2010. 10.1007/s10846-009-9366-2.

Howard Demuth and Mark Beale. Neural network toolbox users guide. Technical report, The MathWorks, Inc, 2000. URL `www.mathworks.com`.

H. Deng, J. Wang, P. Liu, C. Zhou, and J. Wu. Simulation system design of a small-scale unmanned helicopter. *International Journal of Modelling, Identification and Control*, 12(1/2):6–11, 2011.

Linse Dennis and Robert F. Stengel. Identification of aerodynamic coefficients using computational neural networks. In *Archive Set 448*, Meeting Paper Archive, Irvine, CA, 1992. American Institute of Aeronautics and Astronautics. doi: 10.2514/6.1992-172.

Jia-Wen Dong, Ji-Xin Qian, and You-Xian Sun. Recurrent neural networks for recursive identification of nonlinear dynamic process. In *Industrial Technology, 1994. Proceedings of the IEEE International Conference on*, pages 794 –798, Dec 1994.

Henri Eisenbeiss. A mini unmanned aerial vehicle (uav): System overview and image acquisition. In Fuse T. Remondino F. Gruen A., Murai Sh., editor, *International Workshop on Processing and Visualization using High-Resolution Imagery*, Pitsanulok, Thailand, November 2004. International Society for Photogrammetry and Remote Sensing, CIPA.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 1551-6709. doi: 10.1207/s15516709cog1402_1. URL `http://dx.doi.org/10.1207/s15516709cog1402_1`.

William E. Faller and Scott J. Schreck. Neural networks: Applications and opportunities in aeronautics. *Progress in Aerospace Sciences*, 32(5):433 – 456, 1996. ISSN 0376-0421. doi: 10.1016/0376-0421(95)00011-9. URL `http://www.sciencedirect.com/science/article/pii/0376042195000119`.

R. Fletcher. *Practical methods of optimization*. Wiley, Chichester; New York, 2nd edition, 1981.

Ken-Ichi Funahashi. On the approximate realization of continuous mappings by neural networks. *Neural Networks*, 2(3):183 – 192, 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90003-8. URL `http://www.sciencedirect.com/science/article/pii/0893608089900038`.

Matthew Garratt and Sreenatha Anavatti. Non-linear control of heave for an unmanned helicopter using a neural network. *Journal of Intelligent & Robotic Systems*, 66:495–504, 2012. ISSN 0921-0296. doi: 10.1007/s10846-011-9634-9. URL `http://dx.doi.org/10.1007/s10846-011-9634-9`.

Jorge L. Garriga and Masoud Soroush. Model predictive control tuning methods: A review. *Industrial & Engineering Chemistry Research*, 49(8):3505–3515, 2010. doi: 10.1021/ie900323c. URL `http://pubs.acs.org/doi/abs/10.1021/ie900323c`.

V. Gavrilets, B. Mettler, and E. Feron. Nonlinear model for a small-size acrobatic helicopter. In *AIAA Guidance, Navigation, and Control Conference*, number AIAA-2001-4333. AIAA, 2001.

Vladislav Gavrilets, Massachusetts Institute of Technology. Dept. of Aeronautics, and Astronautics. *Autonomous aerobatic maneuvering of miniature helicopters*. Phd thesis, Dept. of Aeronautics and Astronautics, Massachusetts Institute of Technology, 2003.

M. Gopinathan, J.D. Boskovic, R.K. Mehra, and Constantino Rago. A multiple model predictive scheme for fault-tolerant flight control design. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, volume 2, pages 1376–1381 vol.2, 1998. doi: 10.1109/CDC.1998.758477.

J. Grauer, J. Conroy, J. Hubbard, J. Humbert, and D. Pines. System identification of a miniature helicopter. *Journal of Aircraft*, 46(4):1260–1269, 2009.

M.T. Hagan and H.B. Demuth. Neural networks for control. In *American Control Conference, 1999. Proceedings of the 1999*, volume 3, pages 1642 –1656, 1999. doi: 10.1109/ACC.1999.786109.

Simon S. Haykin. *Neural networks and learning machines*. Prentice Hall, New York, 3rd edition, 2009.

Robert K. Heffley and Marc A. Mnich. Minimum-complexity helicopter simulation math model. Technical report, National Aeronautics and Space Administration, Ames Research Center, US Army Aviation Systems Command, 1988. Distributed to depository libraries in microfiche. Accessed from http://nla.gov.au/nla.cat-vn4077617.

Bill G Horne and C Lee Giles. An experimental comparison of recurrent neural networks. In G Tesauro, D Touretzky, and T Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 697–704. The MIT Press, 1995.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366, 1989. ISSN 0893-6080. doi: 10.1016/0893-6080(89)90020-8. URL `http://www.sciencedirect.com/science/article/pii/0893608089900208`.

D. Hunter and B. Wilamowski. Parallel multi-layer neural network architecture with improved efficiency. In *Human System Interactions (HSI), 2011 4th International Conference on*, pages 299–304, 2011.

Jarmo Ilonen, Joni-Kristian Kamarainen, and Jouni Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17 (1):93–105, 2003. ISSN 1370-4621. doi: 10.1023/A:1022995128597. URL `http://dx.doi.org/10.1023/A%3A1022995128597`.

Alberto Isidori. *Nonlinear control systems*. Communications and control engineering series. Springer, Berlin ; New York, 3 edition, 1995.

Wang Jianguo, Zhang Wenxing, Qin Bo, and Shi Wei. Regression rules extraction from artificial neural network based on least squares. In *Natural Computation (ICNC), 2011 Seventh International Conference on*, volume 1, pages 203 –207, July 2011. doi: 10.1109/ICNC.2011.6021906.

E. Joelianto, E. M. Sumarjono, A. Budiyono, and D. R. Penggalih. Model predictive control for autonomous unmanned helicopters. *Aircraft Engineering and Aerospace Technology*, 83(6):375–387, 2011.

Wayne Johnson. *Helicopter theory*. Princeton University Press, Princeton, 1980.

Adem Kalinli and Seref Sagiroglu. Elman network with embedded memory for system identification. *J. Inf. Sci. Eng.*, 22(6):1555–1568, 2006.

S. M. Kamruzzaman and Md M. Islam. Extraction of symbolic rules from artificial neural networks. *WASET Transactions on Science, Engineering and Technology*, 10: 271–277, 2010. URL `http://arxiv.org/abs/1009.4570`.

Bekir Karlik and A. Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122, 2010.

Farid Kendoul. Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems. *Journal of Field Robotics*, 29(2):315–378, 2012. ISSN 1556-4967.

A. Khosravi, S. Nahavandi, D. Creighton, and A.F. Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *Neural Networks, IEEE Transactions on*, 22(9):1341 –1356, Sept. 2011. ISSN 1045-9227. doi: 10.1109/TNN.2011.2162110.

S. K. Kim and D. M. Tilbury. Mathematical modeling and experimental identification of an unmanned helicopter robot with flybar dynamics. *J. Robot. Syst.*, 21(3):95–116, 2004.

Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *The 1995 International Joint Conference on Artificial Intelligence*, pages 1137–1145, 1995.

M. Vijaya Kumar, S. Suresh, S.N. Omkar, Ranjan Ganguli, and Prasad Sampath. A direct adaptive neural command controller design for an unstable helicopter. *Engineering Applications of Artificial Intelligence*, 22(2):181 – 191, 2009. ISSN 0952-1976. doi: 10.1016/j.engappai.2008.07.004. URL `http://www.sciencedirect.com/science/article/pii/S0952197608001280`.

MV Kumar, SN Omkar, R. Ganguli, P. Sampath, and S. Suresh. Identification of helicopter dynamics using recurrent neural networks and flight data. *JOURNAL OF THE AMERICAN HELICOPTER SOCIETY*, 51(2):164–174, 2006.

Rajan Kumar, Ranjan Ganguli, and S.N. Omkar. Rotorcraft parameter estimation using radial basis function neural network. *Applied Mathematics and Computation*, 216(2):584 – 597, 2010. ISSN 0096-3003. doi: 10.1016/j.amc.2010.01.081. URL `http://www.sciencedirect.com/science/article/pii/S0096300310001050`.

A.T. Kutay, A.J. Calise, M. Idan, and N. Hovakimyan. Experimental results on adaptive output feedback control using a laboratory model helicopter. *Control Systems Technology, IEEE Transactions on*, 13(2):196 – 202, march 2005. ISSN 1063-6536. doi: 10.1109/TCST.2004.839563.

M. Kuure-Kinsey and B.W. Bequette. Improved nonlinear predictive control performance using recurrent neural networks. In *American Control Conference, 2008*, pages 4197–4202, 2008. doi: 10.1109/ACC.2008.4587152.

M. Kuure-Kinsey, R. Cutright, and B.W. Bequette. Computationally efficient neural predictive control based on a feedforward architecture. In *American Control Conference, 2006*, pages 6 pp.–, 2006a. doi: 10.1109/ACC.2006.1657169.

M. Kuure-Kinsey, R. Cutright, and BW Bequette. Computationally efficient neural predictive control based on a feedforward architecture. *Industrial & Engineering Chemistry Research*, 45(25):8575–8582, 2006b.

M. Lawrynczuk. A family of model predictive control algorithms with artificial neural networks. *International Journal of Applied Mathematics and Computer Science*, 17 (2):217–232, 2007a. 288ZN Times Cited:17 Cited References Count:42.

M. Lawrynczuk. An efficient nonlinear predictive control algorithm with neural models based on multipoint on-line linearisation. In *EUROCON, 2007. The International Conference on "Computer as a Tool"*, pages 777 –784, September 2007b. doi: 10.1109/EURCON.2007.4400364.

Tsungnan Lin, B.G. Horne, P. Tino, and C.L. Giles. Learning long-term dependencies in narx recurrent neural networks. *Neural Networks, IEEE Transactions on*, 7(6): 1329 –1338, Nov 1996. ISSN 1045-9227.

Lennart Ljung. *System identification : theory for the user*. Prentice Hall information and system sciences series. Prentice Hall PTR, Upper Saddle River, N.J., 2nd edition, 1999.

Lennart Ljung and Torsten Soderstrom. *Theory and practice of recursive identification*. The MIT Press series in signal processing, optimization, and control. MIT Press, Cambridge, Mass., 1983.

David G. Luenberger and Yinyu Ye. *Linear and nonlinear programming*, volume 116 of *International series in operations research & management science*. Springer, New York, 3 edition, 2008. ISBN 0387745025, 9780387745022, 0387745033, 9780387745039.

Jan Marian Maciejowski. *Predictive control : with constraints*. Prentice Hall, Harlow, England, 2002.

M. Y. Mashor. Hybrid multilayered perceptron networks. *International Journal of Systems Science*, 31(6):771–785, 2000.

M. Y. Mashor. Performance comparison between hmlp, mlp and rbf networks with application to on-line system identification. In *Cybernetics and Intelligent Systems, 2004 IEEE Conference on*, volume 1, pages 643–648, 2004.

M.Y. Mashor. Modified recursive prediction error algorithm for training layered neural network. *International Journal of the Computer, the Internet and Management*, 11 (2):24–36, 2003.

R. May, G. Dandy, and H. Maier. *Review of Input Variable Selection Methods for Artificial Neural Networks*, chapter 2, pages 19–44. InTech, 2011.

D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789 – 814, 2000. ISSN 0005-1098. doi: 10.1016/S0005-1098(99)00214-9. URL `http://www.sciencedirect.com/science/article/pii/S0005109899002149`.

B. Mettler, Mark B. Tischler, and T. Kanade. System identification of a model-scale helicopter. Technical report, Robotics Institute, Carnegie Mellon University, 2000.

B. Mettler, C. Dever, and E. Feron. Identification modeling, flying qualities, and dynamic scaling of miniature rotorcraft. In *NATO System Concepts and Integration (SCI) Symposium*, Berlin, May 2002a.

Bernard Mettler. *Identification modeling and characteristics of miniature rotorcraft*. Kluwer Academic Publishers, Boston, 1st edition, 2003.

Bernard Mettler, Mark B. Tischler, and Takeo Kanade. System identification modeling of a small-scale unmanned rotorcraft for flight control design. *Journal of the American Helicopter Society*, 47(1):50–63, 2002b.

A. Naghdinezhad, M. Mohamadian, and A. Dastfan. Comparison of off line neural network training methods for sensorless induction motor drive. In *Universities Power Engineering Conference, 2006. UPEC '06. Proceedings of the 41st International*, volume 2, pages 709 –713, Sept. 2006. doi: 10.1109/UPEC.2006.367571.

H. Nakanishi and K. Inoue. Development of autonomous flight control systems for unmanned helicopter by use of neural networks. In *Neural Networks, 2002. IJCNN '02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2626 –2631, 2002. doi: 10.1109/IJCNN.2002.1007558.

H. Nakanishi, H. Hashimoto, N. Hosokawa, A. Sato, and K. Inoue. Autonomous flight control system for unmanned helicopter using neural networks. In *SICE 2002. Proceedings of the 41st SICE Annual Conference*, volume 2, pages 777 – 782 vol.2, aug. 2002. doi: 10.1109/SICE.2002.1195255.

K.S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *Neural Networks, IEEE Transactions on*, 1(1):4 –27, mar 1990. ISSN 1045-9227. doi: 10.1109/72.80202.

L. S. H. Ngia and J. Sjoberg. Efficient training of neural nets for nonlinear adaptive filtering using a recursive levenberg-marquardt algorithm. *IEEE Transactions on Signal Processing*, 48(7):1915–1927, 2000.

David Nodland, Arpita Ghosh, H. Zargarzadeh, and S. Jagannathan. Neuro-optimal control of an unmanned helicopter. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 2012. doi: 10.1177/1548512912450369. URL `http://dms.sagepub.com/content/early/2012/07/15/1548512912450369.abstract`.

K. Nonami. *Autonomous flying robots: unmanned aerial vehicles and micro aerial vehicles*. Springer, New York, 2010. ISBN 4431538550, 9784431538561, 4431538569, 9784431538554.

Kenzo Nonami, Farid Kendoul, Satoshi Suzuki, Wei Wang, and Daisuke Nakazawa. Linearization and identification of helicopter model for hierarchical control design. In *Autonomous Flying Robots*, pages 95–130. Springer Japan, 2010. ISBN 978-4-431-53855-4. doi: 10.1007/978-4-431-53856-1_5. URL `http://dx.doi.org/10.1007/978-4-431-53856-1_5`.

M. Norgaard, P. H. Sorensen, N. K. Poulsen, O. Ravn, and L. K. Hansen. Intelligent predictive control of nonlinear processes using neural networks. In *Intelligent Control, 1996., Proceedings of the 1996 IEEE International Symposium on*, pages 301–306, 1996.

Magnus Norgaard. *Neural networks for modelling and control of dynamic systems : a practitioner's handbook*. Advanced textbooks in control and signal processing. Springer, Berlin ; New York, 2nd edition, 2000.

Tokunbo Ogunfunmi. *Adaptive nonlinear system identification : the Volterra and Wiener model approaches*. Signals and communication technology. Springer, New York, 2007.

Andrzej W. Ordys and David W. Clarke. A state-space description for gpc controllers. *International Journal of Systems Science*, 24(9):1727–1744, 1993. URL `http://www.tandfonline.com/doi/abs/10.1080/00207729308949590`.

G. D. Padfield. *Helicopter flight dynamics : the theory and application of flying qualities and simulation modelling*. American Institute of Aeronautics and Astronautics, Washington, DC, 2nd edition, 2007.

Mukta Paliwal and Usha A. Kumar. Neural networks and statistical techniques: A review of applications. *Expert Systems with Applications*, 36(1):2 – 17, 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2007.10.005. URL `http://www.sciencedirect.com/science/article/pii/S0957417407004952`.

B. E. Parker, Jr., Todd M. Nigro, Monica P. Carley, Roger L. Barron, David G. Ward, H. V. Poor, Dennis Rock, and Thomas A. DuBois. Helicopter gearbox diagnostics and prognostics using vibration signature analysis. In *Applications of Artificial Neural Networks IV*, volume 1965, pages 531–542, Orlandi, FL, 1993. SPIE-The International Society for Optical Engineering. doi: 10.1117/12.152553. URL `+http://dx.doi.org/10.1117/12.152553`.

J.O. Pedro and P. Kantue. Online aerodynamic parameter estimation of a miniature unmanned helicopter using radial basis function neural networks. In *Control Conference (ASCC), 2011 8th Asian*, pages 1170 –1175, May 2011.

D. T. Pham and X. Liu. Training of elman networks and dynamic system modelling. *International Journal of Systems Science*, 27(2):221–226, 1996.

D.T. Pham and X. Liu. Identification of linear and nonlinear dynamic systems using recurrent neural networks. *Artificial Intelligence in Engineering*, 8(1):67 – 75, 1993. ISSN 0954-1810.

Martin Pottmann and Dale E. Seborg. A nonlinear predictive control strategy based on radial basis function models. *Computers and Chemical Engineering*, 21(9):965–980, 1997.

Raymond W. Prouty. *Helicopter performance, stability, and control*. PWS Engineering, Boston, 1986.

I. E. Putro, A. Budiyono, K. J. Yoon, and D. H. Kim. Modeling of unmanned small scale rotorcraft based on neural network identification. In *2008 IEEE International Conference on Robotics and Biomimetics, Vols 1-4*, pages 1938–1943, 2009.

V. R. Puttige and S. G. Anavatti. Real-time neural network based online identification technique for a uav platform. In *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, pages 92–92, 2006.

Vishwas Ramadas Puttige. *Neural network based adaptive control for autonomous flight of fixed wing unmanned aerial vehicles*. PhD thesis, School of Aerospace, Civil and Mechanical Engineering, Australian Defence Force Academy, University of New South Wales, August 2009. URL `http://handle.unsw.edu.au/1959.4/43736`.

A. Rahideh, M.H. Shaheed, and H.J.C. Huijberts. Dynamic modelling of a trms using analytical and empirical approaches. *Control Engineering Practice*, 16(3):241 – 259, 2008. ISSN 0967-0661.

J. B. Rawlings. Tutorial overview of model predictive control. *Control Systems Magazine, IEEE*, 20(3):38–52, 2000.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, volume 1, pages 586 –591, 1993.

B.P. Rimal, I.E. Putro, A. Budiyono, D. Min, and E. Choi. *System Identification of NN-based Model Reference Control of RUAV during Hover*, chapter 19, pages 395–420. InTech, 2011.

Roadmap. Unmanned aircraft systems roadmap, 2005-2030. Technical report, Office of the Secretary of Defense, Department of Defense, USA, 2005.

J. A. Rossiter. *Model-based predictive control : a practical approach*. CRC Press, Boca Raton, 2003.

S. Saarinen, R. Bramley, and G. Cybenko. Ill-conditioning in neural network training problems. *SIAM J. Sci. Comput.*, 14(3):693–714, May 1993. ISSN 1064-8275. doi: 10.1137/0914044. URL `http://dx.doi.org/10.1137/0914044`.

Balasaheb Patre Sadhana Chidrawar and Laxman Waghmare. *Neural Generalized Predictive Control for Industrial Processes*, chapter 12, pages 199–230. InTech, 2009. doi: 10.5772/7905. URL `http://www.intechopen.com/books/automation-control-theory-and-practice/neural-generalized-predictive-control-for-industrial-processes`.

Mario E. Salgado, Graham C. Goodwin, and Richard H. Middleton. Modified least squares algorithm incorporating exponential resetting and forgetting. *International Journal of Control*, 47(2):477–491, 1988. URL `http://www.tandfonline.com/doi/abs/10.1080/00207178808906026`.

M. K. Samal, S. Anavatti, and M. Garratt. Real-time neural network based identification of a rotary-wing uav dynamics for autonomous flight. In *Industrial Technology, 2009. ICIT 2009. IEEE International Conference on*, pages 1–6, 2009.

Mahendra Samal. *Neural network based identification and control of an unmanned helicopter*. Phd thesis, School of Engineering and Information Technology, University of New South Wales, 2009.

Mahendra Kumar Samal, Sreenatha Anavatti, and Matthew Garratt. Neural network based system identification for autonomous flight of an eagle helicopter. In *17th World Congress The International Federation of Automatic Control*, volume 17, pages 7421–7426, 2008.

Mahendra Kumar Samal, Sreenatha Anavatti, Tapabrata Ray, and Matthew Garratt. A computationally efficient approach for nn based system identification of a rotary wing uav. *International Journal of Control, Automation and Systems*, 8:727–734, 2010. ISSN 1598-6446. doi: 10.1007/s12555-010-0403-5. URL `http://dx.doi.org/10.1007/s12555-010-0403-5`.

Sandhya Samarasinghe. *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. Auerbach Publications, Boca Raton, FL, 1 edition, 2007. ISBN 084933375X, 9780849333750.

R. San Martin, A. Barrientos, P. Gutierrez, and J. del Cerro. Unmanned aerial vehicle (uav) modelling based on supervised neural networks. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2497 –2502, May 2006.

C. P. Sanders, P. A. DeBitetto, E. Feron, H. F. Vuong, and N. Leveson. Hierarchical control of small autonomous helicopters. In *Decision and Control, 1998. Proceedings of the 37th IEEE Conference on*, volume 4, pages 3629–3634, 1998.

S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. Visually guided landing of an unmanned aerial vehicle. *Robotics and Automation, IEEE Transactions on*, 19(3): 371–380, 2003.

P.S. Sastry, G. Santharam, and K.P. Unnikrishnan. Memory neuron networks for identification and control of dynamical systems. *Neural Networks, IEEE Transactions on*, 5(2):306 –319, mar 1994. ISSN 1045-9227. doi: 10.1109/72.279193.

R. Setiono, Wee Kheng Leow, and J.M. Zurada. Extraction of rules from artificial neural networks for nonlinear regression. *Neural Networks, IEEE Transactions on*, 13 (3):564 –577, May 2002. ISSN 1045-9227. doi: 10.1109/TNN.2002.1000125.

M. Hasan Shaheed. Feedforward neural network based non-linear dynamic modelling of a trms using rprop algorithm. *Aircraft Engineering and Aerospace Technology*, 77(1): 13–22, 2005.

M.H. Shaheed and M.O. Tokhi. Dynamic modelling of a single-link flexible manipulator: parametric and non-parametric approaches. *Robotica*, 20(01):93–109, 2002.

Syariful S. Shamsudin and XiaoQi Chen. Identification of an unmanned helicopter system using optimised neural network structure. *International Journal of Modelling, Identification and Control*, 17(3):223–241, 2012a.

Syariful Syafiq Shamsudin and XiaoQi Chen. Recursive gauss-newton based training algorithm for neural network modelling of an unmanned helicopter dynamics. In *Mechatronics and Machine Vision in Practice (M2VIP), 2012 19th International Conference*, pages 92–99, 2012b.

D.H. Shim. *Hierarchical Flight Control System Synthesis for Rotorcraft based Unmanned Aerial Vehicles*. Phd thesis, University of California, Berkeley, 2000.

Jinok Shin, Kenzo Nonami, Daigo Fujiwara, and Kensaku Hazawa. Model-based optimal attitude and positioning control of small-scale unmanned helicopter. *Robotica*, 23(01): 51–63, 2005.

R. Shridhar and D.J. Cooper. A tuning strategy for unconstrained siso model predictive control. *Industrial and Engineering Chemistry Research*, 36(3): 729–746, 1997. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-0031102426&partnerID=40&md5=6063700f6de461b3fc3e66184d45cd7f`.

J. Sjoberg and L. Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62(6):1391–1407, 1995. doi: 10.1080/00207179508921605. URL `http://www.tandfonline.com/doi/abs/10.1080/00207179508921605`.

J. J. E. Slotine and Weiping Li. *Applied nonlinear control*. Prentice Hall, Englewood Cliffs, N.J., 1991.

Ronald Soeterboek. *Predictive control : a unified approach*. Prentice Hall, New York, 1992.

Ariela Sofer, Igor Griva, and Stephen Nash. *Linear and nonlinear optimization*. Society for Industrial and Applied Mathematics, Philadelphia, 2nd edition, 2009. ISBN 9780898716610, 0898716616.

Donald Soloway and P.J. Haley. Neural generalized predictive control. In *Intelligent Control, 1996., Proceedings of the 1996 IEEE International Symposium on*, pages 277–282, 1996. doi: 10.1109/ISIC.1996.556214.

L. Sragner and G. Horvath. Improved model order estimation for nonlinear dynamic systems. In *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2003. Proceedings of the Second IEEE International Workshop on*, pages 266 –271, September 2003.

M. Stinchcombe and H. White. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions. In *Neural Networks, 1989. IJCNN., International Joint Conference on*, volume 1, pages 613 –617, 1989. doi: 10.1109/IJCNN.1989.118640.

Bidyadhar Subudhi and Debashisha Jena. Differential evolution and levenberg marquardt trained neural network scheme for nonlinear system identification. *Neural Processing Letters*, 27(3):285–296, 2008. ISSN 1370-4621. doi: 10.1007/s11063-008-9077-x. URL `http://dx.doi.org/10.1007/s11063-008-9077-x`.

Bidyadhar Subudhi and Debashisha Jena. A differential evolution based neural network approach to nonlinear system identification. *Applied Soft Computing*, 11(1):861 – 871, 2011. ISSN 1568-4946. doi: 10.1016/j.asoc.2010.01.006. URL `http://www.sciencedirect.com/science/article/pii/S1568494610000116`.

S. Suresh and N. Sundararajan. An on-line learning neural controller for helicopters performing highly nonlinear maneuvers. *Applied Soft Computing*, 12(1):360 – 371, 2012.

ISSN 1568-4946. doi: 10.1016/j.asoc.2011.08.036. URL http://www.sciencedirect.com/science/article/pii/S156849461100319X.

S. Suresh, M. V. Kumar, S. N. Omkar, V. Mani, and P. Sampath. Neural networks based identification of helicopter dynamics using flight data. In *9th International Conference on Neural Information Processing ICONIP (2002)*, volume 1, pages 10–14, 2002.

S. Suresh, S.N. Omkar, V. Mani, and T.N. Guru Prakash. Lift coefficient prediction at high angle of attack using recurrent neural network. *Aerospace Science and Technology*, 7(8):595 – 602, 2003. ISSN 1270-9638.

Z. Taha, A. Deboucha, and M. Bin Dahari. Small-scale helicopter system identification model using recurrent neural networks. In *TENCON 2010 - 2010 IEEE Region 10 Conference*, pages 1393–1397, 2010.

Mark B. Tischler and Robert K. Remple. *Aircraft and rotorcraft system identification : engineering methods with flight-test examples*. AIAA education series. American Institute of Aeronautics and Astronautics, Reston, VA, 2006.

Quan Truong. Continuous-time model predictive control. Master's thesis, School of Electrical and Computer Engineering, RMIT University, 2007.

Jack V. Tu. Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. *Journal of Clinical Epidemiology*, 49(11):1225 – 1231, 1996. ISSN 0895-4356.

Sr. Urnes, J., R. Davidson, and S. Jacobson. A damage adaptive flight control system using neural network technology. In *American Control Conference, 2001. Proceedings of the 2001*, volume 4, pages 2907 – 2912, 2001. doi: 10.1109/ACC.2001.946344.

K. Valavanis. *Advances in unmanned aerial vehicles: state of the art and the road to autonomy*, volume 33. Springer, Dordrecht, 2007.

M. Valenti, B. Bethke, G. Fiore, J. P. How, and E. Feron. Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Keystone, CO, August 2006. URL http://acl.mit.edu/papers/GNC06_ValentiHow.pdf.

M. Valenti, B. Bethke, D. Dale, A. Frank, J. McGrew, S. Ahrens, J.P. How, and J. Vian. The mit indoor multi-vehicle flight testbed. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2758 –2759, april 2007. doi: 10.1109/ROBOT.2007.363882.

M. Vijaya Kumar, P. Sampath, S. Suresh, S. N. Omkar, and Ranjan Ganguli. Neural network based feedback error controller for helicopter. *Aircraft Engineering and Aerospace Technology*, 83(5):283–295, 2011. URL http://search.proquest.com.ezproxy.canterbury.ac.nz/docview/893410432?accountid=14499.

J.C. Avila Vilchis, B. Brogliato, A. Dzul, and R. Lozano. Nonlinear modelling and control of helicopters. *Automatica*, 39(9):1583 – 1596, 2003. ISSN 0005-1098. doi: 10.1016/S0005-1098(03)00168-7. URL http://www.sciencedirect.com/science/article/pii/S0005109803001687.

Nikos Vitzilaios and Nikos Tsourveloudis. An experimental test bed for small unmanned helicopters. *Journal of Intelligent & Robotic Systems*, 54(5):769–794, 2009. 10.1007/s10846-008-9284-8.

D.J. Walker. Multivariable control of the longitudinal and lateral dynamics of a fly-by-wire helicopter. *Control Engineering Practice*, 11(7):781 – 795, 2003. ISSN 0967-0661. doi: 10.1016/S0967-0661(02)00189-2. URL `http://www.sciencedirect.com/science/article/pii/S0967066102001892`.

J. Wang, Z. Bai, and P. Liu. Modelling and control system design of a small-scale unmanned helicopter. *International Journal of Modelling, Identification and Control*, 12(1/2):12–16, 2011.

Liuping Wang. *Model predictive control system design and implementation using MATLAB*. Springer, London, 2009a. ISBN 1848823312, 9781849968362, 9781848823310, 1849968365, 9781848823303, 1848823304.

Liuping Wang. Classical mpc systems in state-space formulation. In *Model Predictive Control System Design and Implementation Using MATLAB*, Advances in Industrial Control, pages 297–332. Springer London, 2009b. ISBN 978-1-84882-331-0. URL `http://dx.doi.org/10.1007/978-1-84882-331-0_9`.

Liuping Wang. Discrete-time mpc for beginners. In *Model Predictive Control System Design and Implementation Using MATLAB*, Advances in Industrial Control, pages 1–42. Springer London, 2009c. ISBN 978-1-84882-330-3. URL `http://dx.doi.org/10.1007/978-1-84882-331-0_1`.

Liuping Wang. Discrete-time mpc with constraints. In *Model Predictive Control System Design and Implementation Using MATLAB*, Advances in Industrial Control, pages 43–84. Springer London, 2009d. ISBN 978-1-84882-330-3. URL `http://dx.doi.org/10.1007/978-1-84882-331-0_2`.

Liuping Wang and Peter C. Young. An improved structure for model predictive control using non-minimal state space realisation. *Journal of Process Control*, 16(4):355 – 371, 2006. ISSN 0959-1524. doi: http://dx.doi.org/10.1016/j.jprocont.2005.06.016. URL `http://www.sciencedirect.com/science/article/pii/S0959152405000910`.

Y. Wang, D. J. Miller, and R. Clarke. Approaches to working in high-dimensional data spaces: gene expression microarrays. *British Journal of Cancer*, 98(6):1023–1028, 2008.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *1960 IRE WESCON Convention Record*, volume 4, pages 96–104, New York, 1960. IRE. URL `http://isl-www.stanford.edu/~{}widrow/papers/c1960adaptiveswitching.pdf`.

B. M. Wilamowski. Neural network architectures and learning algorithms. *Industrial Electronics Magazine, IEEE*, 3(4):56–63, 2009.

B. M. Wilamowski and Yu Hao. Improved computation for levenberg-marquardt training. *Neural Networks, IEEE Transactions on*, 21(6):930–937, 2010.

B. M. Wilamowski, N. J. Cotton, O. Kaynak, and G. Dundar. Computing gradient vector and jacobian matrix in arbitrarily connected neural networks. *Industrial Electronics, IEEE Transactions on*, 55(10):3784–3790, 2008.

Bogdan Wilamowski. *Neural Networks Learning*, chapter 11, pages 1–18. CRC Press, 2011a.

Bogdan Wilamowski. *Understanding Neural Networks*, pages 1–11. CRC Press, 2011b. 2011/04/11 0 doi:10.1201/b10604-8.

Bogdan Wilamowski, Hao Yu, and Nicholas Cotton. *NBN Algorithm*, chapter 13, pages 1–24. CRC Press, 2011.

J. Witt, S. Boonto, and H. Werner. Approximate model predictive control of a 3-dof helicopter. In *Decision and Control, 2007 46th IEEE Conference on*, pages 4501–4506, 2007.

Zhang Youmin and X. R. Li. A fast u-d factorization-based learning algorithm with applications to nonlinear system modeling and identification. *IEEE Transactions on Neural Networks*, 10(4):930–938, 1999.

Hao Yu and Bogdan Wilamowski. *Levenberg-Marquardt Training*, chapter 12, pages 1–16. CRC Press, 2011.